Master's thesis
Master's Programme in Data Science

Paragraph-length image captioning using hierarchical recurrent neural networks

Arturs Polis

March 29, 2019

Supervisor(s): Patrik Floréen and Mats Sjöberg

Examiner(s): Patrik Floréen
Petri Myllymäki

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Arturs Polis

Työn nimi — Arbetets titel — Title

Paragraph-length image captioning using hierarchical recurrent neural networks

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	March 29, 2019	83

Tiivistelmä — Referat — Abstract

Recently, a neural network based approach to automatic generation of image descriptions has become popular. Originally introduced as *neural image captioning*, it refers to a family of models where several neural network components are connected end-to-end to infer the most likely caption given an input image. Neural image captioning models usually comprise a Convolutional Neural Network (CNN) based image encoder and a Recurrent Neural Network (RNN) language model for generating image captions based on the output of the CNN.

Generating long image captions – commonly referred to as *paragraph captions* – is more challenging than producing shorter, sentence-length captions. When generating paragraph captions, the model has more degrees of freedom, due to a larger total number of combinations of possible sentences that can be produced. In this thesis, we describe a combination of two approaches to improve paragraph captioning: using a hierarchical RNN model that adds a top-level RNN to keep track of the sentence context, and using richer visual features obtained from dense captioning networks. In addition to the standard MS-COCO Captions dataset used for image captioning, we also utilize the Stanford-Paragraph dataset specifically designed for paragraph captioning.

This thesis describes experiments performed on three variants of RNNs for generating paragraph captions. The *flat* model uses a non-hierarchical RNN, the *hierarchical* model implements a two-level, hierarchical RNN, and the *hierarchical-coherent* model improves the *hierarchical* model by optimizing the coherence between sentences.

In the experiments, the *flat* model outperforms the published non-hierarchical baseline and reaches similar results to our *hierarchical* model. The *hierarchical* model performs similarly to the corresponding published model, thus validating it. The *hierarchical-coherent* model gives us inconclusive results – it outperforms our *hierarchical* model but does not reach the same scores as the corresponding published model.

With our *flat* model implementation, we have shown that with minor improvements to a simple image captioning model, one can obtain much higher scores on standard metrics than previously reported. However, it is yet unclear whether a hierarchical RNN is required to model the paragraph captions, or whether a single RNN layer on its own can be powerful enough. Our initial human evaluation indicates that the captions produced by a hierarchical RNN may in fact be more fluent, however the standard automatic evaluation metrics do not capture this.

Avainsanat — Nyckelord — Keywords

neural networks, image captioning, paragraph captioning, hierarchical RNN

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

During recent years, the area of image captioning [67], where one generates short, usually around ten words long, captions to images based on the image content has witnessed a neural network based renaissance [3, 26, 38, 66, 71]. With the advent of GPU-based training, large datasets such as MS-COCO [10, 37], containing more than 100,000 captioned images, deep learning techniques for image captioning have become practical.

With the successful application of neural networks for creating short image captions, research has progressed towards longer, paragraph-length image captions, with 5 - 6 sentences, each of them around a dozen words long [30]. A longer description of an image may be beneficial in various tasks such as image retrieval, video transcription [73], as well as other tasks that require automatic systems to reason about images.

This thesis explores three variations of Encoder-Decoder (defined in Section 2.1) neural network architectures for generating paragraph-length image captions. First, we will describe the baseline image captioning model in Chapter 2. In Chapter 3 we will see how longer image descriptions can be created by applying one more level of hierarchy to the baseline image captioning model. Then, in Chapter 4 we will talk about the experiments in which we used these models for creating different-sized captions. Our conclusions will be presented in Chapter 5.

As we shall see in Sections 2.3.3 and 3.2.1, the implementations for basic building blocks used for generating image descriptions that are either short *captions* or long *paragraphs* are similar. We will refer to "image captioning" when talking about the task of generating short, single-sentence captions, and "paragraph captioning" when talking about task of generating longer, multi-sentence captions. Furthermore, we shall refer to models consisting of a single level of the Recurrent Neural Network (RNN) [18, pp. 367–415] hierarchy as "flat", and models containing multiple levels of RNN hierarchy as "hierarchical".

Evaluating machine-generated captions is challenging. The problem becomes worse with increased length of captions. Several metrics have been proposed to measure the quality of the generated captions. In addition, human evaluation can be used to assess caption quality. In our experiments, we will use the popular BLEU [48], Meteor [6], and CIDEr [65]

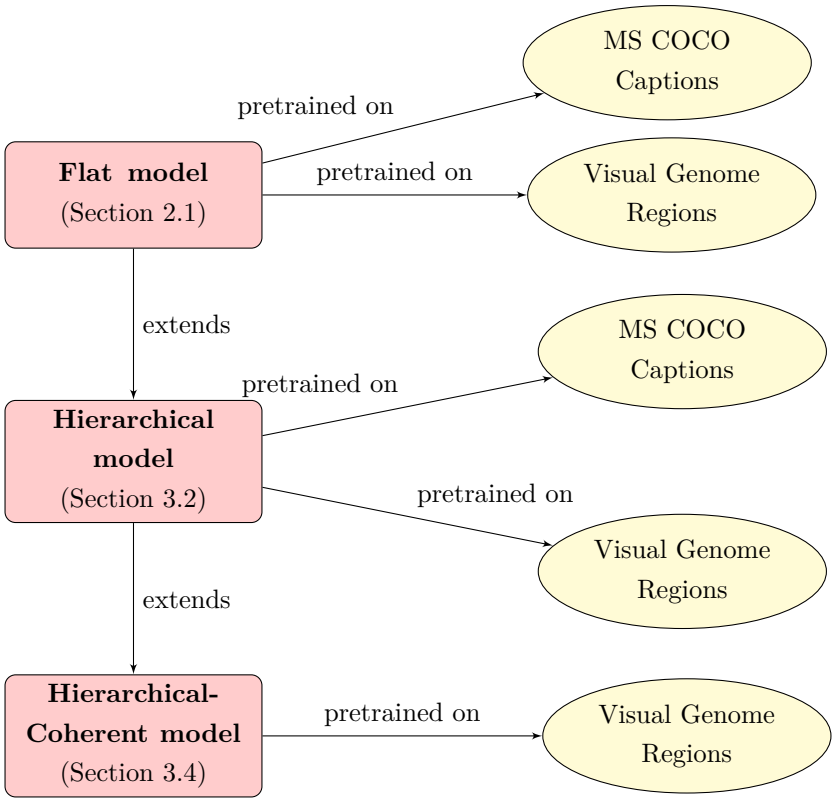


Figure 1.1: Paragraph captioning models covered in this thesis.

metrics for automatic caption evaluation. We shall also look at several different human evaluation criteria in Section 4.2.

Our experiments were performed using the PyTorch* neural network programming framework in the Python 3.5 programming language. In Section 4.4.2 we will describe the experiments using a flat RNN for generating an entire paragraph using a single recurrent cell, while in Sections 4.4.3 and 4.4.4 we will look at paragraph captioning done by a hierarchical RNN with two recurrent network levels, where the first RNN is used to generate sentence context and the second one uses the context to learn individual *tokens*, which are either words or punctuation marks.

The relation between different models covered in our experiments in Chapter 4 is illustrated in Figure 1.1. We train three different models to generate paragraph captions: flat, hierarchical and hierarchical-coherent. The flat model is the basic image captioning model with a decoder component that uses only one RNN layer. The hierarchical model adds an extra RNN level to the flat decoder. Finally, the hierarchical-coherent model introduces a so called "coherence" connection between the two layers of the RNN hierarchy. We pre-train our models using either the MS COCO Captions [10] or the Visual Genome Regions [31] datasets.

*https://pytorch.org/

2. Image Captioning

The task of *image captioning* is defined as the automatic generation of short descriptions of image content using natural language [67]. Recently the use of neural networks for this task has become popular [26, 66, 71]. Figure 2.1 shows a few captions generated by our image captioning model.

A group of people riding bikes down a street.

A woman sitting at a table with a pizza in front of her.

A living room with a television, a chair, and a table.

Figure 2.1: Examples of image captioning output.

This chapter will describe the baseline image captioning model. We will refer to it as the *flat* model, to contrast it with the hierarchical models we describe later. The chapter is organized in the following way: Section 2.1 will explain the principles of the Encoder-Decoder architecture and the language model used for image captioning, Section 2.2 will describe how the input features are prepared, and Section 2.3 will go into the details of the decoder component. The full image captioning pipeline is described in Section 2.4. We conclude the chapter with Section 2.5 taking a brief look at recently proposed extensions to the baseline captioning model.

2.1 Encoder-Decoder Model

On a high level, neural image captioning models are based on an *Encoder-Decoder* architecture [11]. This architectural approach borrows from neural sequence-to-sequence

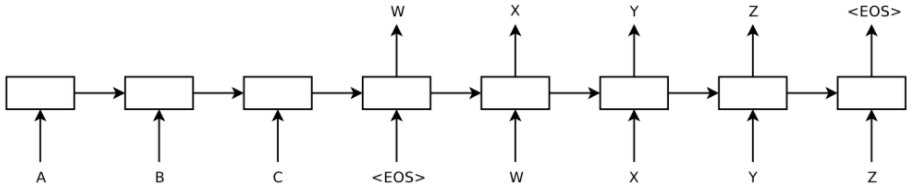


Figure 2.2: A simple sequence-to-sequence model. Input sequence is "ABC", output sequence is "WXYZ" [63].

models [63] in machine translation. Intuitively, captioning can be thought of as translating from image to text. The analogy with machine translation enables the use of methods from text-to-text translation. Figure 2.2 shows a conceptual representation of a model that takes the input sequence "ABC" and learns to predict an output sequence "WXYZ", conditioned on the first sequence.

Typically, the sequence-to-sequence models are implemented using two separate subnetworks. Figure 2.3 shows one such model comprising two separate recurrent neural networks, one network encoding the input and the other predicting the output given the input. These two components are:

- *Encoder* – processes the input, for example a phrase in the source language or an image, once this is done, the encoder outputs *context* c;
- *Decoder* – separate network which takes context c as its input and produces the desired output.

The benefit of a neural Encoder-Decoder architecture is that it is composed of separate well-defined building blocks, while at the same time it is possible to train it end-to-end using backpropagation. The context can be made available to each time-step of the decoder as shown in Figure 2.3, or alternatively it can be fed to the decoder at the initial time-step only. In the case of hierarchical model implementations in Sections 3.2.1 and 3.4.1, the image context is fed to the top-level RNN at each time-step. In our flat model, in

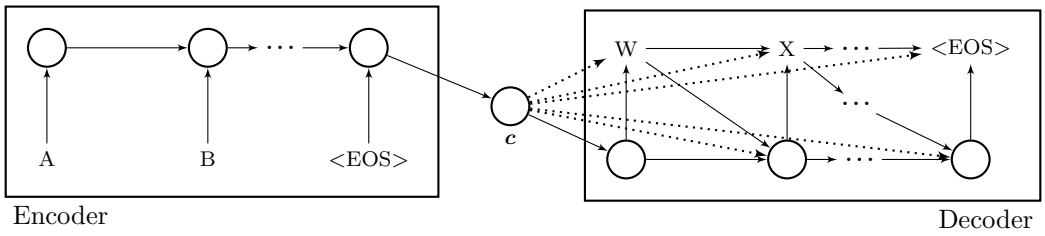


Figure 2.3: The Encoder-Decoder architecture, adapted from Cho et al. [11].

Section 2.3.3, the context c is only fed at the initial step of the recurrence, but feeding the context at each time-step is also possible [58].

Vinyals et al. [66] applied the Encoder-Decoder architecture to image caption generation. They proposed a captioning model that established a baseline for neural Encoder-Decoder image captioning models. The model consists of a convolutional neural network (CNN) [18, pp. 326–366] encoder, that encodes the image into a vector representation and a RNN decoder that "unpacks" the vector representation into a human readable description by implementing a language model described below.

2.1.1 Image Captioning Language Model

Let us introduce the concepts of *vocabulary* and *language model*. The vocabulary is a pre-defined and ordered collection of size V, of possible tokens available to the model for text generation. Each token w_i in the vocabulary is stored at the corresponding index position i. Also recall, that each token can represent either a single word or a punctuation character. When speaking about image captioning, one can use the term "word" to refer to both the individual words as well as punctuation characters.

A language model in its most simple form defines a probability distribution over all the tokens w_i in our vocabulary such that:

$$\sum_{i=1}^{V} p(w_i) = 1. \tag{2.1}$$

Language models become useful for generating the sequences of tokens, once we introduce the ability to condition the probability distribution of the next token w_t on the sequence of previous tokens $w_1, \ldots, w_{t-1}$. Such a model allows us to use the chain rule of probability to obtain the joint probability of a token sequence of length N:

$$p_t(w_t|w_{t-1}, \ldots, w_1) = p_{t-1}(w_{t-1}|w_{t-2}, \ldots, w_1)p_{t-2}(w_{t-2}|w_{t-3}, \ldots, w_1) \ldots p_1(w_1), \tag{2.2}$$

$$p(w_t, \ldots, w_1) = \prod_{t=1}^{N} p_t(w_t|w_{t-1}, \ldots, w_1). \tag{2.3}$$

When training language models, the aim is that the model would assign higher probabilities to word orders more likely to occur in natural language, for example:

$$p(\text{two giraffes grazing in the field}) > p(\text{giraffes two grazing in the field}). \tag{2.4}$$

A *conditional language model* is defined in the same way, except that all tokens in the sequence share the same conditioning context c:

$$p(w_t, \ldots, w_1|c) = \prod_{t=1}^{N} p_t(w_t|c, w_{t-1}, \ldots, w_1). \tag{2.5}$$

Let $(w_1, \dots w_N)$ be a sequence of tokens representing a caption of length N, with words that are not in the vocabulary represented by a special $<UNK>$ token. Each token is also mapped to its *one-hot* representation $\boldsymbol{S_t}$, a vector of size V with all other elements set to zero except for the one at the index position corresponding to the token w_t in the vocabulary. Thus, the sequence $\boldsymbol{S} = (\boldsymbol{S_1}, \dots, \boldsymbol{S_N})$ is the caption encoded as N one-hot vectors.

The task of image captioning is to learn a conditional language model, where the probability of the next token being generated depends on both the vectorized representation of the input image $\boldsymbol{I}$, as well as the previous tokens in the sequence:

$$p(\boldsymbol{S}|\boldsymbol{I}) = p(\boldsymbol{S_N}, \dots, \boldsymbol{S_1}|\boldsymbol{I}) = \prod_{t=1}^{N} p_t(\boldsymbol{S_t}|\boldsymbol{I}, \boldsymbol{S_{t-1}}, \dots, \boldsymbol{S_1}). \tag{2.6}$$

Vinyals et al. [66] proposed the following objective function, optimizing trainable parameters $\boldsymbol{\Theta}$ of the model, with $\boldsymbol{\Theta}^*$ being the optimal parameter values:

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} \sum_{(\boldsymbol{I},\boldsymbol{S})} \log p(\boldsymbol{S}|\boldsymbol{I}; \boldsymbol{\Theta}). \tag{2.7}$$

Equation 2.7 represents a conditional language model in logarithmic probability space. Here, $\boldsymbol{S}$ is the ground truth caption and $\boldsymbol{I}$ the input image. The conditional probability can be modeled using the chain rule, where the probability of the token $\boldsymbol{S_t}$ generated at position t in the caption $\boldsymbol{S}$, containing N tokens, is conditioned on the set of tokens in positions $1, \dots, t-1$ and the input image. Thus, the joint probability of all tokens in the sequence $\boldsymbol{S}$ is modeled by the following equation:

$$\log p(\boldsymbol{S}|\boldsymbol{I}; \boldsymbol{\Theta}) = \sum_{t=1}^{N} \log p(\boldsymbol{S_t}|\boldsymbol{I}, \boldsymbol{S_{t-1}}, \dots, \boldsymbol{S_1}; \boldsymbol{\Theta}). \tag{2.8}$$

The model uses *teacher forcing* [18, p. 372], which means feeding the ground truth word $\boldsymbol{S_t}$ to the model at each time-step during training. At inference stage, teacher forcing cannot be used, and the model is conditioned on its own output tokens $\hat{\boldsymbol{S}}_{\boldsymbol{t}}$ at each inference time-step:

$$\log p(\boldsymbol{S}|\boldsymbol{I}; \boldsymbol{\Theta}) = \sum_{t=1}^{N} \log p(\boldsymbol{S_t}|\boldsymbol{I}, \hat{\boldsymbol{S}}_{\boldsymbol{t-1}}, \dots, \hat{\boldsymbol{S}}_{\boldsymbol{1}}; \boldsymbol{\Theta}), \tag{2.9}$$

where $\hat{\boldsymbol{S}}_{\boldsymbol{t-1}}, \dots, \hat{\boldsymbol{S}}_{\boldsymbol{1}}$ are the best guesses by the model at each time-step preceding t.

2.2 Encoder: Image Features Extractor

In order to condition the language model on an image, the image needs to be converted into a suitable representation. Such a representation is known as an *image feature*. The process of obtaining image features is called *feature extraction*. The aspects of the image that we want to encode into features depend on the task. Features can be low-level – and represent the visual properties of the image – color, texture, local contrast/edges, shapes (lines, points, circles). Extracting such features may not even require a neural network based approach. The high-level, or semantic features [2], encode properties of the image in terms of objects and their relationships to one another.

The use of convolutional networks pre-trained on an image classification task have emerged as a viable solution for semantic feature extraction for use in image captioning. Such networks are typically pre-trained on the ImageNet [56] image classification task, where for each input image the trained model needs to predict the most likely class selected from 1000 alternatives. The phenomenon that makes it possible to use neural networks trained on one task in another task is called *transfer learning* [47]. Thanks to transfer learning, CNNs trained on ImageNet can be used to provide input features to models trained for tasks other than just classification. It has been shown [72] that early layers of a network learn generic, lower level features, which may be useful for many tasks, while deeper layers learn more specialized and abstract features. When applying transfer learning, an output from a layer of appropriate depth is picked for further use.

Several image classification networks have emerged recently, such as AlexNet [32], VGG16 and VGG19 [59], GoogLeNet (also known as Inception network) [64], and ResNet [23]. ResNet stands for Residual Network. It has recently gained popularity as an image feature extraction model for image captioning [3, 38]. The ResNet architecture was designed to address the problems of *vanishing*, as well as *exploding* gradients [8, 17].

Vanishing gradients may happen as the number of layers in the network grows. If the absolute value of each gradient is close to 0, backpropagating through the network is bound to result in close to zero gradients for early layers in the net. Such gradients are said to be vanishing, because they stop affecting learning. Having too small gradients means that it will take a network a long time to converge in the best case scenario, while in the worst case scenario the learning can stall before converging.

The ResNet architecture seeks to primarily address the vanishing gradient problem. The ResNet model is built from a number of residual blocks, one of which is shown in Figure 2.4. Each residual block consists of a sub-network and a "shortcut connection" acting as an identity function between the current layer and the layer preceding the residual

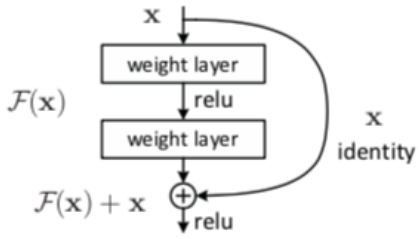


Figure 2.4: A residual block in the ResNet architecture [23].

block. In the figure there are two intermediate layers that learn some mapping $\mathcal{F}(\boldsymbol{x})$, and the shortcut connection adds $\boldsymbol{x}$ to the output of the intermediate layers. This helps to ensure that gradients flow to lower layers by skipping intermediate layers, thus helping the network to learn at least the identity function between lower and higher layers.

Conversely, if the gradients are much larger than 1.0, multiplying them across many layers results in early layers having exploding gradients that are close to infinity, making weight updates impossible. The authors of the ResNet architecture claim that the residual blocks also help to avoid exploding gradients [23].

There exists a number of ResNet variants with different number of layers. Popular variants are ResNet-34, ResNet-50, ResNet-101, ResNet-152 and ResNet-200, where the latter part of the name indicates total number of weighted layers. Following recent work in image captioning [38], we chose to perform experiments using the ResNet-152 architecture. Figure 2.5 shows the full network diagram for a smaller ResNet variant with a total of 34 layers. The architecture of this smaller network follows the same principles as the 152-layer version we are using. For comparison, the ResNet network is shown alongside the VGG19 network and a 34-layer convolutional net without shortcut connections.

Features for image captioning are extracted from ResNet-152 by removing the last classification layer that outputs the probabilities of the image being in one of the 1000 classes. The remaining last layer outputs an average-pooled vector of dimension $D_{ResNet} = 2048$. We use it as our image feature vector. Before supplying the extracted feature vector to the decoder, the vector is mapped to the decoder input dimension via matrix multiplication. In the following section we provide more details on how we use the ResNet-152 network for image feature extraction.

2.2.1 ResNet Feature Extraction

Before extracting ResNet-152 features, we preprocess each image. First we resize each image to a uniform size of 256×256 pixels. Next, the RGB channels of each 3-dimensional image pixel $\boldsymbol{x}$ are normalized using the following predetermined per-channel averages $\boldsymbol{\mu}$

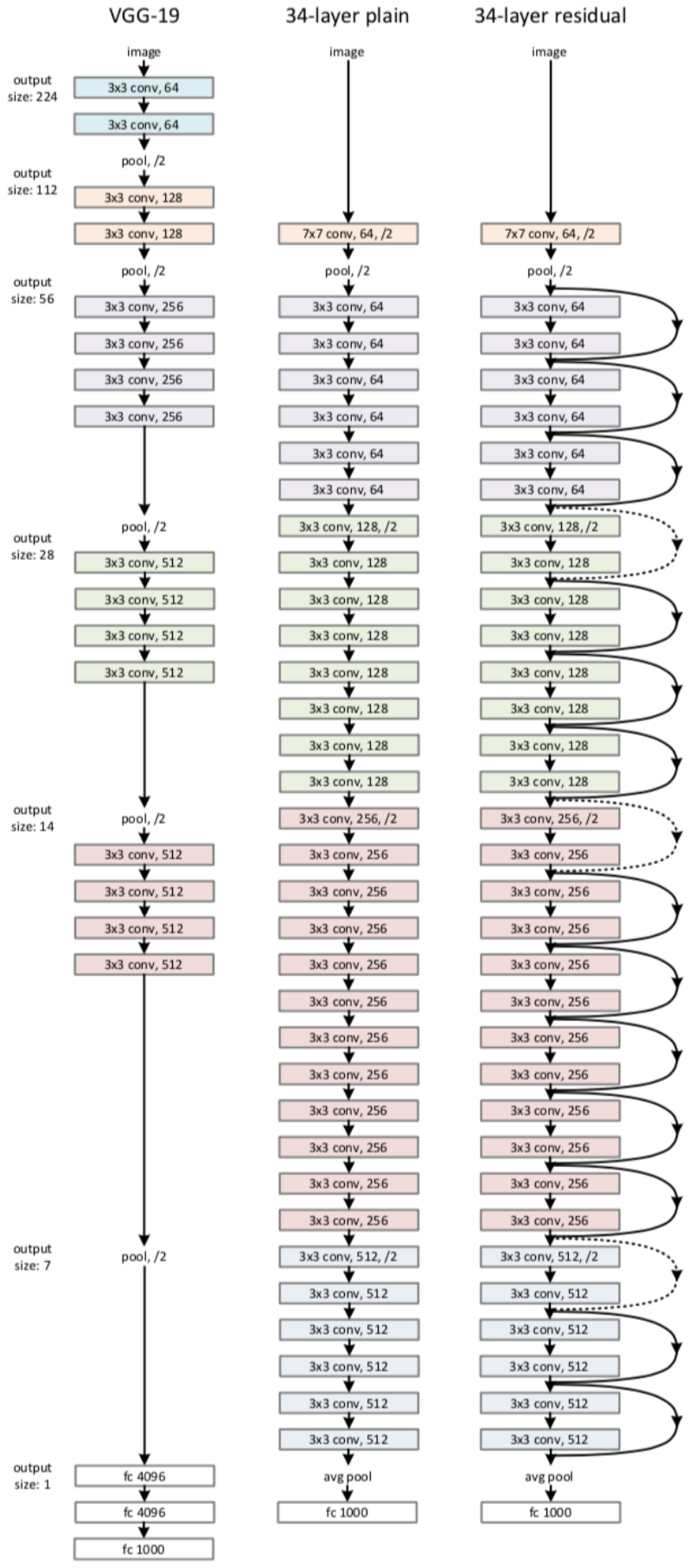


Figure 2.5: ResNet-34 architecture (right), VGG19 (left), and 34-layer network without shortcut connections (center) [23].

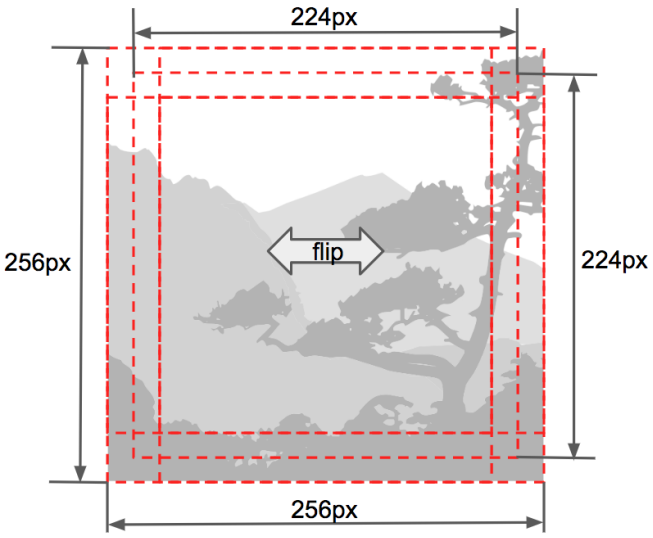


Figure 2.6: Ten-crop of an image is created by cropping five 224×224 regions from an 256×256 image and horizontally flipping each cropped region.

and standard deviations $\boldsymbol{\sigma}$:

$$\boldsymbol{\mu} = (0.485, 0.456, 0.406) \tag{2.10}$$

$$\boldsymbol{\sigma} = (0.229, 0.224, 0.225) \tag{2.11}$$

$$\boldsymbol{x}_{normalized} = \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}. \tag{2.12}$$

The normalization values for the channel means and standard deviations are given in the PyTorch documentation*, and are calculated from ImageNet [56] data. This normalization step is required because the ResNet-152 model is pre-trained on images normalized in the same way.

Once we have obtained the normalized pixel values for the 256×256 input images, five 224×224 sized image regions are cropped – with four of the cropped images aligned at each of the respective corners of the original image, and the last one taken from the center, as shown in Figure 2.6. Furthermore, each of the crops is flipped horizontally, making up a total of ten cropped images. PyTorch provides a built-in function called *ten-crop* for performing this operation in one step.

After the ten-crop images are created, each of them is passed through a pre-trained ResNet-152 model. We then keep the output of the second last layer of the pre-trained model, of size $D_{ResNet} = 2048$ for each of the input crops. There are several ways to combine the outputs for ten-crop images into one. We experiment with two strategies:

*https://pytorch.org/docs/stable/torchvision/models.html

Table 2.1: The MS COCO captioning task results using ten-crop and random crop features computed from ResNet-152.

Features	**CIDEr**	**Meteor**
Single random crop	85.18	23.93
Ten-crop	87.17	24.26

calculating either an element-wise mean of the ten resulting vectors – referred to as *average pooling*, or taking element-wise maximum – referred to as *maximum pooling*. The pooled output features are pre-calculated and stored to disk before training, so that they can be retrieved on-demand without costly computation.

The pooled features retain the same dimension, D_{ResNet} as the features obtained from just one input image without ten-crop, but at the same time they provide better empirical results on image captioning task, as shown in Table 2.1. Here we can see that when using ten-crop features the model outperforms the simple data-augmentation strategy of random cropping each image at each iteration as measured using CIDEr [65] and Meteor [6] scores (see Section 4.2). The pre-calculated ten-crop features speed up each pass over the data, providing more bandwidth for trials and experimentation. Random cropping, on the other hand, requires the full ResNet-152 CNN to be run for each image at every epoch, which slows the training significantly.

2.3 Decoder: Image Caption Generator

The image features input to the decoder are mapped to a vector of size E, expected to contain crucial information about the source image, and in fact can be thought of as a crudely compressed abstract representation of the image content [72]. The objective of the decoder is to learn to condition on this abstract representation of an image, and generate a caption that adequately describes the content of the source image.

2.3.1 Shared Representation for Words and Images

The conditioning of the language model on the E-dimensional image context vector $\boldsymbol{c}$ is possible largely due to learning a shared representation [62] of images and sentences and the use of the recurrent network-based language model [40]. The shared representation allows one to encode image features and individual words in the caption using the same latent space. Recurrent network based language models, on the other hand, allow us to model the probability distributions of particular sequences of words given an input image.

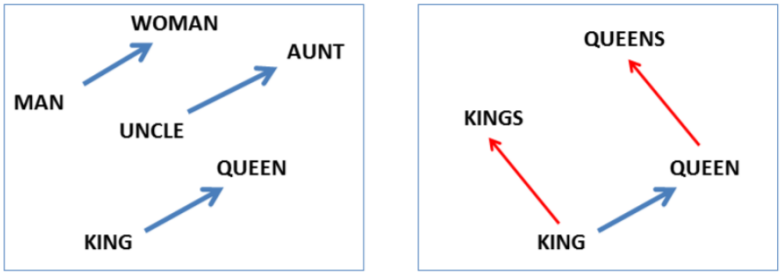


Figure 2.7: Lower dimensional projection of two types of relations: singular-to-plural and male-to-female [42].

In order to be able to train a neural language model, Bengio et al. [7] proposed the idea of encoding each word in a vocabulary using a distributed, continuous representation, instead of the traditional "bag-of-words" approach, where each word is kept as a discrete 0 or 1 indicator at a given place in a one-hot vector of size V. Given a vocabulary of size $V = 10000$ words, in bag-of-words, our input size for each word in a caption would be a long and sparse vector of mostly zeros. On the other hand, if one would manage to transform this sparse representation into continuous space, training neural network models with such transformed inputs would become possible.

Word embeddings provide such a representation. Each word in the vocabulary is mapped to an E-dimensional, continuous embedding, where $E \ll V$ and E is the size of the encoder output. Now, instead of mapping each word to a discrete index, we have a continuous "neighborhood" of words, where similar words are often neighbors in an E-dimensional embedding space. This is due to the way the embeddings are constructed – for example, if two words occur in similar contexts in the training data, they are more likely to occupy nearby locations in the embedding space [41]. Relations between similar concepts may often be interpreted as vectors in embedding space [42] when projected to lower dimensions, as shown in Figure 2.7. Due to the still quite high dimensionality of the embedding, many such relations can be in principle contained at once inside each individual embedding [42, 62].

At this point we are still missing a shared representation between words and images. Shared representation between individual words and images depends on the notion of *compositional semantics* [44], where each embedding vector can in fact represent either a word or an entire phrase. Compositional distributed representation of an entire sentence can be used together with image features to learn a *multi-modal* [62] representation space where the compositional image caption representation maps to the same vector space as image features. The ability to obtain a shared representation [26] for multi-modal input is at the very heart of both image captioning covered in this chapter, as well as hierarchical paragraph captioning covered in Chapter 3, where the RNNs at different levels receive

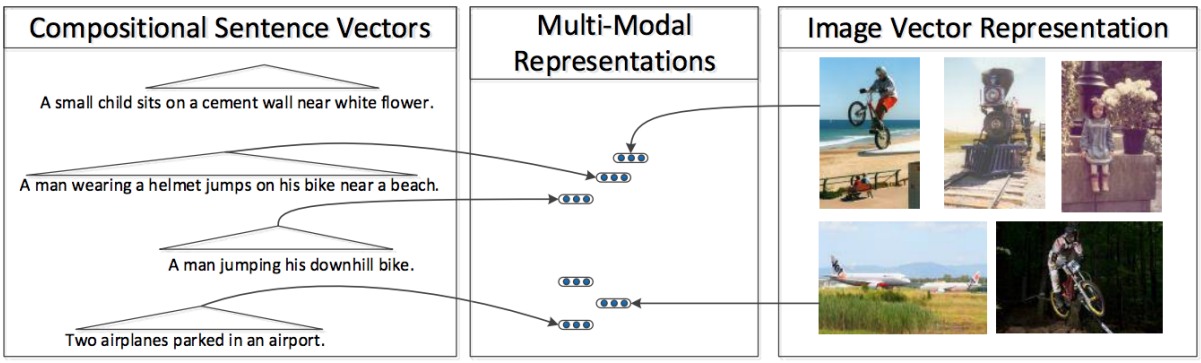


Figure 2.8: Sentences and images mapping to a shared vector representation [62].

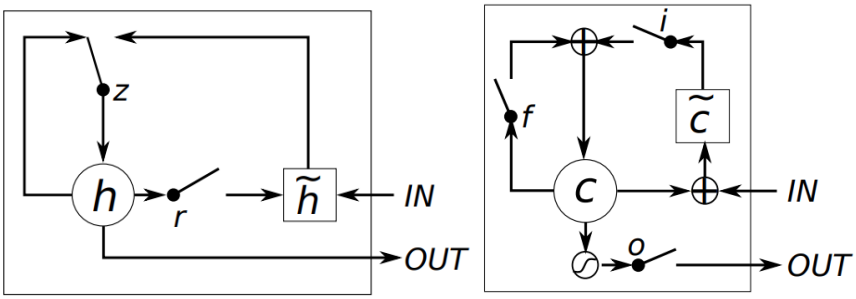
different modalities as their input. Figure 2.8 shows image feature vectors and sentence vectors representing two different input modalities mapped to a shared embedding space.

It is possible to use pre-trained word embeddings when training captioning models, and many reusable embeddings exist, such as Word2Vec [39], Glove [50], and more recently ELMO [51] and Bert [15]. However, it is also possible to train the embeddings jointly with the rest of the model, which is the strategy we chose in the experiments covered in this thesis.

2.3.2 Recurrent Neural Language Models

A key component in image captioning is the recurrent neural language model [40]. Such a language model is trained to predict the next word in the sequence given all the previous words and a context vector. Recurrent neural nets are a diverse and powerful class of models for generating both discrete and continuous valued sequences [20]. In the case of text generation, the RNN takes a previously generated token as its input and thus conditions the next token to be generated on its previous outputs. In order to be able to generalize to previously unseen inputs and generate novel, yet semantically correct sequences, the RNN needs to learn to interpolate between the training examples it sees.

Just like convolutional networks, RNNs may suffer from vanishing gradients, thus essentially "forgetting" the inputs that came before a certain point in the past. In order to be effective in correctly generating long sequences, recurrent nets need to be able to selectively "remember" and "forget" longer-term history depending on its current relevance. Modern RNNs are typically implemented using either Long Short-Term Memory (LSTM) [24] or Gated Recurrent Units (GRU) [11]. Both types of recurrent cells provide mechanisms for selectively keeping longer and shorter term contexts by employing a series of gates. Each gate is a single layer neural network with a sigmoid $\sigma(\cdot)$ activation. At time-step t we denote the current element in the input sequence as $\boldsymbol{x_t}$, and we subscript



(a) GRU Cell **(b)** LSTM Cell

Figure 2.9: Internal structure of GRU and LSTM recurrent units [12].

the name of the gate with t to indicate the output from the gate.

GRU and LSTM cells, shown in Figure 2.9, are different in terms of how many gates they have, and how they store and output their internal state.

A **GRU unit** contains two gates: *reset* and *update*:

- *Reset gate* – r determines how much of the old hidden state vector of the RNN cell $\boldsymbol{h_{t-1}}$ to use for the new hidden state proposal vector $\tilde{\boldsymbol{h}}_t$:

$$\boldsymbol{r_t} = \sigma(\boldsymbol{W_{rx}}\boldsymbol{x_t} + \boldsymbol{W_{rh}}\boldsymbol{h_{t-1}}), \tag{2.13}$$

  where $\boldsymbol{W_{rx}}, \boldsymbol{W_{rh}}$ are learnable weight matrices, and $\boldsymbol{h_{t-1}}$ the hidden state output from the previous time-step $t-1$. The proposed new hidden state is calculated as follows:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_x}\boldsymbol{x_t} + \boldsymbol{W_h}(\boldsymbol{r_t} \odot \boldsymbol{h_{t-1}})), \tag{2.14}$$

  where "$\odot$" is element-wise multiplication.

- *Update gate* – z fulfills a dual role, of both z and $1-z$, its complement. The little "switch" next to gate z seen in Figure 2.9 can be thought of as a dial indicating the mixing proportion between the previous hidden state $\boldsymbol{h_{t-1}}$ and the proposed new hidden state $\tilde{\boldsymbol{h}}_t$.

  If the output of z is close to 0, the cell uses mostly the new hidden state $\tilde{\boldsymbol{h}}_t$. If z is close to 1, the old hidden state $\boldsymbol{h_{t-1}}$ is used. The value of the gate is calculated in the same manner as for the reset gate:

$$\boldsymbol{z_t} = \sigma(\boldsymbol{W_{zx}}\boldsymbol{x_t} + \boldsymbol{W_{zh}}\boldsymbol{h_{t-1}}). \tag{2.15}$$

The new value of the hidden state $\boldsymbol{h_t}$ of the GRU unit is a mixture between the old hidden state $\boldsymbol{h_{t-1}}$ and the new hidden state proposal $\tilde{\boldsymbol{h}}_t$. At the end of any given time-step t,

the hidden state of the GRU unit is thus the same as its output:

$$\boldsymbol{h_t} = (\mathbf{1} - \boldsymbol{z_t}) \odot \boldsymbol{h_{t-1}} + \boldsymbol{z_t} \odot \tilde{\boldsymbol{h}}_t. \tag{2.16}$$

An **LSTM cell** is different from the GRU, because it maintains a separate cell state $\boldsymbol{c_t}$, and has three gates: *input*, *output* and *forget*.

- *Input gate* – i is complementary to the forget gate f. However, unlike in GRU, they are both implemented separately. The input gate is implemented as a sigmoid, and it controls how much the new cell state $\boldsymbol{c_t}$ is comprised of the proposed new cell state $\tilde{\boldsymbol{c}}_t$:

  $$\boldsymbol{i_t} = \sigma(\boldsymbol{W_{ix}}\boldsymbol{x_t} + \boldsymbol{W_{ih}}\boldsymbol{h_{t-1}}). \tag{2.17}$$

  The proposed new cell state $\tilde{\boldsymbol{c}}_t$ is calculated in the following way:

  $$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W_x}\boldsymbol{x_t} + \boldsymbol{W_h}\boldsymbol{h_{t-1}}). \tag{2.18}$$

  Unlike in GRU, cell state vector $\boldsymbol{c_t}$ and the LSTM output vector $\boldsymbol{h_t}$ are different from each other.

- *Forget gate* – f complements the input gate, and controls the weighing for the previous value of cell state $\boldsymbol{c_{t-1}}$:

  $$\boldsymbol{f_t} = \sigma(\boldsymbol{W_{fx}}\boldsymbol{x_t} + \boldsymbol{W_{fh}}\boldsymbol{h_{t-1}}). \tag{2.19}$$

  Using the output of the input and forget gates the LSTM cell calculates the new value for cell state $\boldsymbol{c_t}$:

  $$\boldsymbol{c_t} = \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \odot \tilde{\boldsymbol{c}}_t. \tag{2.20}$$

  Before the LSTM cell can output, one more gate needs to do its work, acting on the output of the cell.

- *Output gate* – o is used for calculating the final output of the LSTM cell:

  $$\boldsymbol{o_t} = \sigma(\boldsymbol{W_{ox}}\boldsymbol{x_t} + \boldsymbol{W_{oh}}\boldsymbol{h_{t-1}}). \tag{2.21}$$

The output of LSTM is:

$$\boldsymbol{h_t} = \boldsymbol{o_t} \odot \tanh(\boldsymbol{c_t}). \tag{2.22}$$

The hyperbolic tangent (tanh) non-linearity maps the values of cell state $\boldsymbol{c_t}$ into the range $(-1, 1)$, and the output gate controls the element-wise $(0, 1)$ intensity when mapping each element of $\tanh(\boldsymbol{c_t})$ to $\boldsymbol{h_t}$.

Before we can connect the entire chain from word embeddings to output, we need one more component, a *softmax* layer that outputs a probability distribution over words. The image captioning model implements this by projecting the RNN output $\boldsymbol{h_t}$ of size H to a vector of size V, which is the same size as the vocabulary, where $\boldsymbol{W_y} \in \mathbb{R}^V \times \mathbb{R}^H$:

$$\boldsymbol{y_t} = \boldsymbol{W_y h_t}. \tag{2.23}$$

For a vector $\boldsymbol{y_t}$ of size V, an element-wise softmax is defined as:

$$\text{softmax}(y_{tj}) = \frac{\exp(y_{tj})}{\sum_{i=1}^{V} \exp(y_{ti})}. \tag{2.24}$$

When using greedy [66] decoding, one picks the word corresponding to the index j of the largest softmax output, which is the approach we follow in our experiments.

2.3.3 Decoder Implementation

We implement a flat decoder, shown in Figure 2.10 using a single RNN component with two hidden layers. Our experiments use either GRU or LSTM recurrent cells. The flat decoder can be used for generating single "sentence-length" captions as well as paragraph-length captions, by treating each input paragraph as a single sequence.

Before we begin training, we wrap the ground truth caption with *<START>* and *<END>* tokens. Next, we embed each input token $\boldsymbol{S_t}$ (represented as one-hot vector) in the prepared ground truth caption to a vector $\boldsymbol{x_t}$ of size E. The visual embedding output by the encoder is fed as the initial input, $\boldsymbol{x_0}$, of the RNN. Word embeddings are trained jointly with the rest of the model parameters.

We use mini-batch gradient descent in training, where each pass over the dataset is split into batches of 128 (image, caption) pairs. To accommodate mini-batch training, where

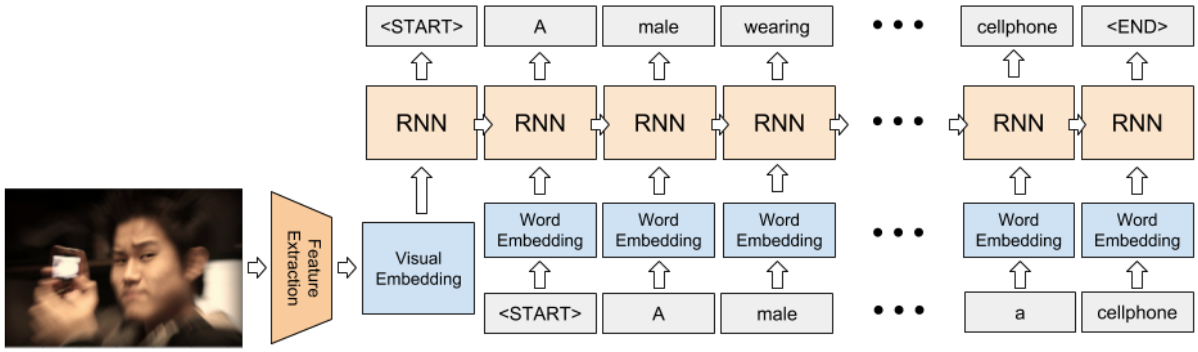


Figure 2.10: Flat decoder.

each caption may be of different length, PyTorch provides the *PaddedSequence* object, which allows training RNNs with mini-batches comprising variable length input sequences. In order to use the PaddedSequence class, each mini-batch needs to be sorted in decreasing order, based on the caption length.

2.4 Image Captioning Pipeline

The full pipeline for the flat model by Vinyals et al. [66] that we use in our experiments is:

$$\boldsymbol{x_0} = \boldsymbol{c} = \text{encoder}(\boldsymbol{I}) \tag{2.25}$$

$$\boldsymbol{h_0} = 0 \tag{2.26}$$

$$\boldsymbol{x_t} = \boldsymbol{W_e}\boldsymbol{S_t}, \quad t \in \{1, \ldots, N\} \tag{2.27}$$

$$\boldsymbol{h_t} = \text{RNN}(\boldsymbol{x_{t-1}}, \boldsymbol{h_{t-1}}), \quad t \in \{1, \ldots, N\} \tag{2.28}$$

$$\boldsymbol{y_t} = \boldsymbol{W_y}\boldsymbol{h_t} \tag{2.29}$$

$$p_t = \text{softmax}(\boldsymbol{y_t}), \quad t \in \{1, \ldots, N\}. \tag{2.30}$$

Where $\boldsymbol{c}$ is the context vector that corresponds to visual embedding of the image features, $\boldsymbol{S_t}$ is the one-hot vector corresponding to the ground truth token at position t in the caption, and $\boldsymbol{W_e} \in \mathbb{R}^E \times \mathbb{R}^V$ is an embedding weights matrix. RNN$(\cdot)$ is either an LSTM or GRU based implementation of a recurrent network. The RNN output $\boldsymbol{h_t}$ is then mapped to a vector $\boldsymbol{y_t}$ of the same size V as the vocabulary.

The loss function $\mathcal{L}$, which the model is trained to minimize, is the sum of negative log-likelihoods of each ground truth token $\boldsymbol{S_t}$, $t \in 1, \ldots, N$:

$$\mathcal{L}(\boldsymbol{I}, \boldsymbol{S}) = -\sum_{t=1}^{N} \log p_t(\boldsymbol{S_t}|\boldsymbol{I}; \boldsymbol{\Theta}). \tag{2.31}$$

In our implementation we use the cross-entropy loss, which combines the softmax output with the negative log-likelihood loss computed in Equation 2.31.

2.5 Proposed Extensions to the Image Captioning

The image captioning model described in preceding sections can be extended. Some of the recently introduced building blocks with potential for improving image captioning performance are:

- *Attention mechanism* [5, 71] seeks to remedy the fact that the single context vector may not be sufficient to contain all the salient details of the input image. Attention allows the decoder to read relevant parts of the input space for each output step, instead of relying on a single context vector [3, 38].

- *Reinforcement learning* (RL) [4, 70] allows to train neural network based models that learn to optimize for discrete, non-differentiable values. Image captioning models that optimize caption scoring metrics are one such example [54], where scores depend on discrete and thus non-differentiable sequences of output words.

- *Generative adversarial networks (GAN)* [19] are a class of models consisting of two main components: a *generator* responsible for creating output proposals given the input, and a *discriminator* which is a network that learns to distinguish outputs coming from the same probability distribution as the ground truth from the ones that are not. GANs have been shown to generate diverse image captions [13, 57] however the captions created using this approach are not that competitive in terms of automatic scores.

- *Variational auto-encoders (VaE)* [28] represent another generative method used in image captioning [53, 68]. The VaE formulation aims to model the probability distribution of the training data, through learning some normally distributed latent representation of the data parameterized by its mean and variance.

Currently the most promising results have been shown by models using attention [3] and reinforcement learning [54].

3. Paragraph Captioning

The task of paragraph captioning and the corresponding training dataset have emerged recently [30]. In traditional image captioning the generated image descriptions are short. For example, the average length of a caption in the MS COCO Captions dataset [10] is just 11 words. Paragraph captions are longer, split into several sentences, and contain 67 words on average. Several ground truth captions from the Stanford-Paragraph dataset [30] are shown in Figure 3.1.

The ground is covered in snow. The snow has foot prints in it. There are four men in the snow. The four men are wearing burgundy shirts, gray pants and black helmets. The four men are riding snowboards. The men in the middle are doing tricks in the air, The man on the left is going down a hill. The man on the far right is doing a trick mid air.

A brown and tan giraffe is walking in a field with branches and dirt on the ground. There is a tree to the left of the giraffe with brown leaves, and bare branches. To the left of the giraffe there is a short, but wide tree with lots of leaves on most of the branches. There is tall grass near the camera in front of the dirt the giraffe is standing on.

A man and woman are sitting at a table in a restaurant. The woman is wearing glasses, a sweater and dark bottoms. She is holding a writing utensil in her hand. The man is staring at a laptop computer that he is typing on. He is wearing eyeglasses, a sweater and a shirt. Another man and woman are sitting on a blue couch against a brick wall working on laptops also.

Figure 3.1: Images and captions from the Stanford-Paragraph dataset [30].

Table 3.1: Different baselines are not directly comparable: Top scores on CIDEr and Meteor metrics for recent image and paragraph captioning models.

Model	**CIDEr**	**Meteor**
Image captioning [3]	117.9	27.6
Paragraph captioning [9]	20.9	18.6
Paragraph captioning – human baseline [30]	28.55	19.22

It may be hard to describe all the relevant aspects of an image in just a dozen words. Bigger, several sentences long, descriptions are often needed. While these longer captions can indeed capture more detail of an image, they are also harder to learn, since as the length of caption increases, the number of places where the machine can make a mistake increases as well. In addition, one cannot compare image and paragraph captioning results to one another by merely looking at the standard metrics. Table 3.1 shows that the baselines are different, and scores obtained by human-generated paragraph captions are significantly lower than the best image captioning results.

The task of paragraph captioning has received less attention compared to image captioning. The ideas emerging from image captioning research, however, are often applicable to paragraph captioning. Therefore there exists a considerable degree of "cross-pollination" between these tasks. Below, we consider the key requirements for when a language model described in Section 2.1.1 is extended from generating simple one-sentence captions to paragraphs containing multiple sentences. In the following two lists we have summarized the requirements for the image and paragraph captioning language models:

Image captioning:

- Models the relations between individual words including meaning and grammar;
- Models the relation between the image and the caption;
- Depends on input features which must capture the main content of the image. Details may sometimes be omitted without affecting caption quality.

Paragraph captioning:

- Models the relations between individual words including meaning and grammar;
- Models the relation between the input image and the entire paragraph;
- Models the relations between the image and each sentence;
- Models the relations between sentences as separate units of meaning – main point of each sentence compared to its neighboring sentences, coherence between sentences;

- Depends on input features that preserve information on the content of different regions of the image.

As we see, the requirements for the two tasks are somewhat similar, yet paragraph captioning introduces the extra requirement for modeling the language on both the sentence and the word level. This means that the input features must contain information that can be used to generate sentences describing different aspects of the image.

The main baseline architecture is the "Regions-Hierarchical" model by Krause et al. [30], which has recently emerged to address the needs of paragraph-length language modeling for describing images. The model introduces an encoder that is pre-trained on a *dense captioning* task, in which each image is captioned with multiple per-region annotations. In addition to the regions-based encoder, the model uses a hierarchical decoder modeling the paragraphs on sentence and word levels separately using two corresponding RNNs. Other models developed for paragraph captioning that have been published recently [9, 30, 35] all adopt variants of this Encoder-Decoder architecture with a hierarchical decoder.

The currently available Stanford-Paragraph dataset [30] has only around 19,000 image caption pairs in total. The dataset contains only one example paragraph per image, compared with the MS-COCO Captions dataset [10], which contains five example captions per each image. For this reason, the paragraph captioning model may suffer from overfitting to the training data, when trained on the Stanford-Paragraph dataset only. One possible solution to this problem is the use of transfer learning, an approach introduced in Section 2.2. Krause et al. [30] propose using two sources of transfer learning – pre-trained image encoder and pre-trained weights for the RNN component responsible for sentence generation. In their work, both the encoder, and the language model were pre-trained on the Visual Genome Regions dataset [31], which we describe in Section 4.1.

In this chapter, Section 3.1 will cover the dense captioning encoder, Section 3.2 will describe the hierarchical decoder, and Section 3.3 will provide the end-to-end picture of the model. The hierarchical-coherent model which extends the hierarchical baseline by aiming to enforce coherence between individual sentences within a paragraph will be introduced in Section 3.4. The chapter concludes with Section 3.5 describing other recent paragraph captioning enhancements.

3.1 Encoder: Dense Captioning Network

Visual features used as input to the hierarchical language model [30] are obtained from a dense captioning network proposed by Johnson et al. [25], also known as *DenseCap*. The dense captioning network combines the tasks of region proposal and image caption-

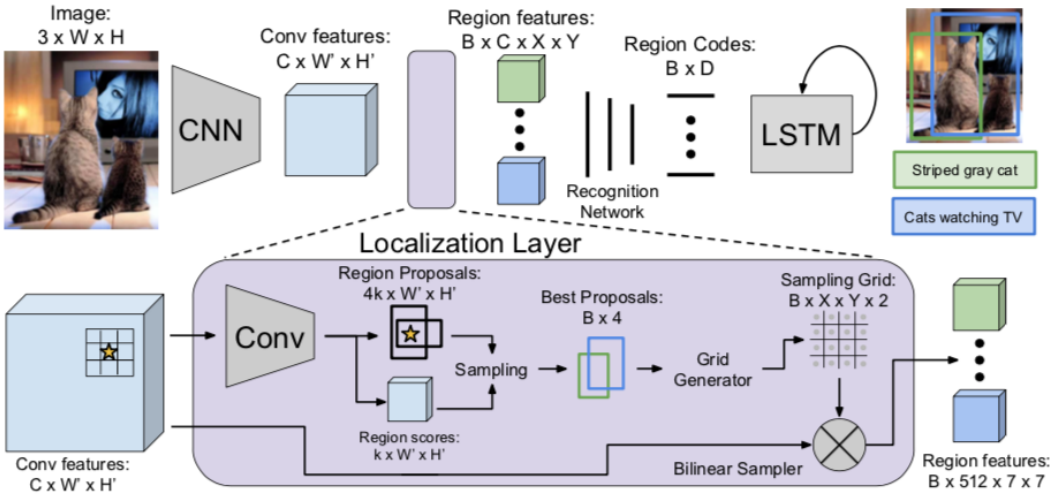


Figure 3.2: DenseCap architecture [25].

ing into a single joint model. The dataset used for training the DenseCap network is Visual Genome Regions [31]. Before providing the details on how DenseCap features are extracted, we will describe some details on how these features are calculated.

In the task of dense captioning, the neural network is trained to output a set of B tuples each containing a bounding box and a caption describing the area of an image bounded by the box. When using the model as an encoder in a larger pipeline, what we are typically interested in is obtaining a feature vector for each salient region of the image.

The dense captioning model is shown in Figure 3.2 and is composed of the following elements:

- Convolutional network for encoding the input image into a $C \times W' \times H'$ convolutional representation;
- Fully convolutional *localization layer* that takes in the convolutional representation of the image and outputs information pertaining to B regions of interest, with B fixed beforehand. The outputs of the layer include: region coordinates, region scores, and region features for each of the detected image regions;
- *Recognition network* takes in the region features and flattens each of them to a vector of size $D_{DenseCap} = 4096$;
- Recurrent neural language model that generates captions for each region in a manner analogous to the one used in image captioning, described in Section 2.1.1.

The dense captioning network also follows the basic Encoder-Decoder pattern introduced in Section 2.1. Unlike in image captioning, the relationship here is "one-to-many" as

each image is responsible for generating B image-region based contexts. Each context is then used as an input to a separate instance of the image captioning task. The encoder in the dense captioning network consists of all the components up to and including the recognition network, the region codes shown in Figure 3.2 act as context vectors, and finally the LSTM-based RNN is the decoder.

The final output of the network is a collection of bounding boxes corresponding to different regions of the input image and natural language descriptions of each such region, as for example shown in Figure 3.3. The loss function is composed of five criterions: $L1$ loss on the region positions and binary logistic loss on the predicted $(0, 1)$ confidences for regions in the localization layer, then again the same two items in the recognition network, and finally cross-entropy on the language model output.

It is important to note, that although it is not strictly a paragraph captioning model, DenseCap can be used to directly create longer image descriptions by concatenating the generated region descriptions together. This approach may generate sentences that are coherent as individual captions for image regions, but they lack coherence when joined together in order to form a paragraph. Therefore, one cannot rely on DenseCap to create plausible paragraphs.

In order to use transfer learning from dense captioning to paragraph captioning, a suitable intermediate representation from DenseCap needs to be identified and used as the context for the decoder. One such suitable representation is the output of the recognition network, which produces input for the region caption generator as seen in Figure 3.2. Because of the way it is used further in the DenseCap pipeline, this representation can be thought of as the compositional embedding (see Section 2.3.1) of each image region.

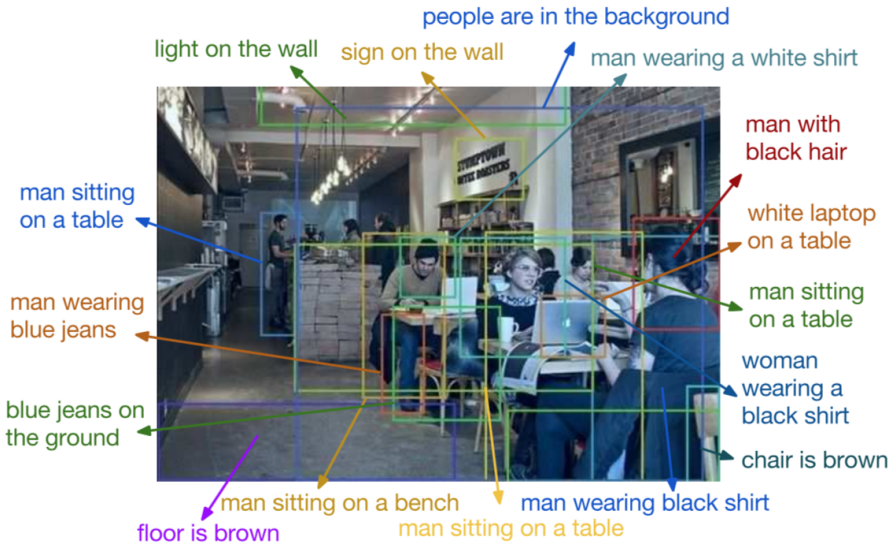


Figure 3.3: Example output of a DenseCap network [25].

3.1.1 Extracting DenseCap Features

From a bird's eye view, the approach of extracting DenseCap features, depicted in Figure 3.4, carries some similarities with the ten-crop feature extraction described in Section 2.2.1. However, there are some substantial differences. Region features obtained from DenseCap are grounded to region content, unlike the ten-crop features which are obtained from the same regions for all input images.

The feature extraction process begins with feeding original-sized images from the dataset to a feature extraction script provided by the DenseCap implementation published by the original authors*. The provided model is trained on a total of 77,398 images containing an average of 50 region captions per image. Before the image data is passed through the network, the model resizes each image in such a way that the longest edge is 720 pixels long. After the model has processed the image, the DenseCap feature extraction script provides us with the following outputs:

- *Detected regions* – a list of B bounding box sizes and coordinates, sorted by salience/importance;
- *Region features* – B per-region feature vectors of size $D_{DenseCap}$, sorted in the same order as the bounding-boxes.

Following published research [9, 30] on the hierarchical models for paragraph generation, we store $B = 50$ region features for each input image, after which we apply maximum pooling by taking element-wise maximum across all B feature vectors, obtaining the final per-image feature-vector of size $D_{DenseCap}$. In addition to maximum pooling, we experimented with average-pooled image features created by calculating an element-wise average of B region vectors.

*https://github.com/jcjohnson/densecap

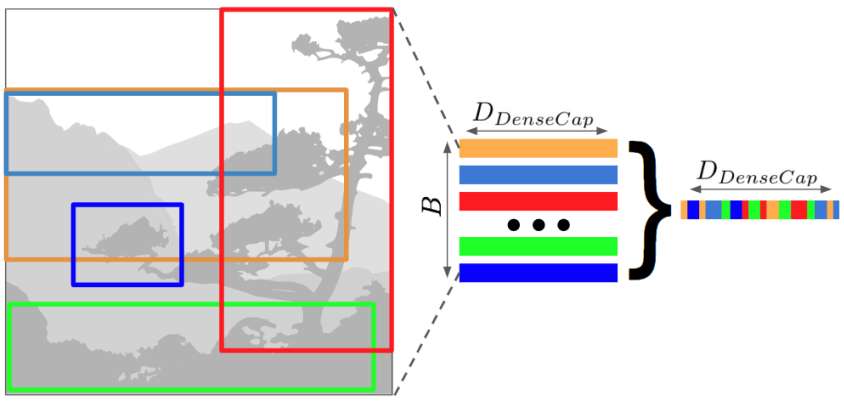


Figure 3.4: DenseCap features extracted from image regions using element-wise maximum.

Finally, the obtained $D_{DenseCap}$-dimensional feature vector is projected to an E-dimensional context vector that is then fed to the hierarchical decoder described in the next section.

3.2 Decoder: Hierarchical RNN

Paragraph captions can be thought to consist of both the individual words as well as the sentences. Li et al. [34] and Lin et al. [36] propose to separately model word-level and sentence-level dependencies by employing a hierarchical language model. A separate RNN would thus model sentences on the word or "token" level, and another would model paragraphs on the sentence level.

The idea of incorporating multiple layers of hierarchy into an RNN based model is not new. El Hihi and Bengio [16] proposed the use of hierarchical models on sequential inputs to aid in modeling longer-term dependencies between the structural units, which are themselves sequential. Ranging from longer-term to shorter-term contexts, in case of text input, these structural units are chapters, paragraphs, sentences, and words. However, with the introduction of Long Short-Term Memory (LSTM), the ability of RNNs to selectively model longer and shorter-term contexts has been improved. Therefore one can also ask whether a modern RNN-based language model does indeed benefit from explicit hierarchy on the level of network architecture. We will try to provide an answer to this question in Chapter 4.

Yu et al. [73] first mentioned the use of a hierarchical language model in the closely related task of video captioning. In their model they leverage a two-level hierarchy, containing word-level and sentence-level RNNs implemented using GRUs. During training, the word-level RNN is initialized from the image features and the sentence context is set to zero. After reading in the first sentence, it passes the average word embedding of all words in that sentence to the sentence-level RNN, which then outputs a new context for the next sentence.

As mentioned earlier, the recent "Regions-Hierarchical" model by Krause et al. [30] has become the de-facto paragraph captioning baseline, and for this reason we will describe their formulation in greater detail. They define the hierarchical RNN used for paragraph captioning as consisting of two levels of hierarchy, each modeled by a separate RNN:

- **SentenceRNN** – Top-level RNN that is responsible for generating a topic vector $\boldsymbol{T_i}$ for each sentence. SentenceRNN takes the same image features output by the encoder as its input at each time-step, and outputs a different topic vector for the next sentence;

- **WordRNN** – Final output-level RNN that generates individual words in each sentence, given the topic vector $\boldsymbol{T_i}$ as its first input. In our experiments this is exactly the same component as the decoder used in the flat model.

In the flat model, each caption is said to be "grounded" [62] directly on the input image $\boldsymbol{I}$. In the hierarchical model the grounding is also hierarchical. The intermediate level of grounding is provided by sentence topics $\boldsymbol{T_i}$, where each topic vector serves the same purpose as did the context vector $\boldsymbol{c}$ in Section 2.4, except that now each sentence in a paragraph has a different context vector.

In the "Regions-Hierarchical" model, the decoder training procedure assumes that we are given a paragraph containing M sentences, and each sentence containing N_i words, with i being the index of the sentence. The decoder is initialized by preloading WordRNN weights from a language model pre-trained on the large Visual Genome Regions region-description dataset [31]. Next, SentenceRNN is run for M time-steps, using the same input at each time-step. The output of the SentenceRNN is a hidden vector of size $H = 512$ which is then passed to two different sub-networks: a *stopping classifier* and a *topic generator*.

The stopping classifier is a logistic layer that converts the hidden layer output to a binary classifier. If the classifier output is below 0.5, the model is in the *CONTINUE* state, meaning that the next sentence can be generated. Otherwise, the *STOP* state has been reached, and no new sentences should be generated.

The topic generator is a two-layer fully connected network. After the stopping classifier has given the model a "go-ahead" to output one more sentence, the topic vector $\boldsymbol{T_i}$ for the next sentence is computed by the topic generator and used as the initial input for WordRNN to start generating the ith sentence. At the level of WordRNN, the captioning process mimics that of the baseline image captioning model described in Section 2.1.1. As in image captioning, we use teacher forcing, providing ground truth words as input at each time-step of the WordRNN.

3.2.1 Hierarchical Decoder Implementation

The hierarchical decoder in Figure 3.5 incorporates the flat decoder, shown in Figure 2.10, as a sub-unit. The hierarchical decoder limits the maximum number of allowed sentences per paragraph to $M_{MAX} = 6$, to make sure that the inference process can terminate. Each sentence in the paragraph is generated by the same WordRNN, using the same weights.

During training, each paragraph is split into sentences. The input to the decoder is the same kind of mapping from image features to embedding size E as used in the flat model.

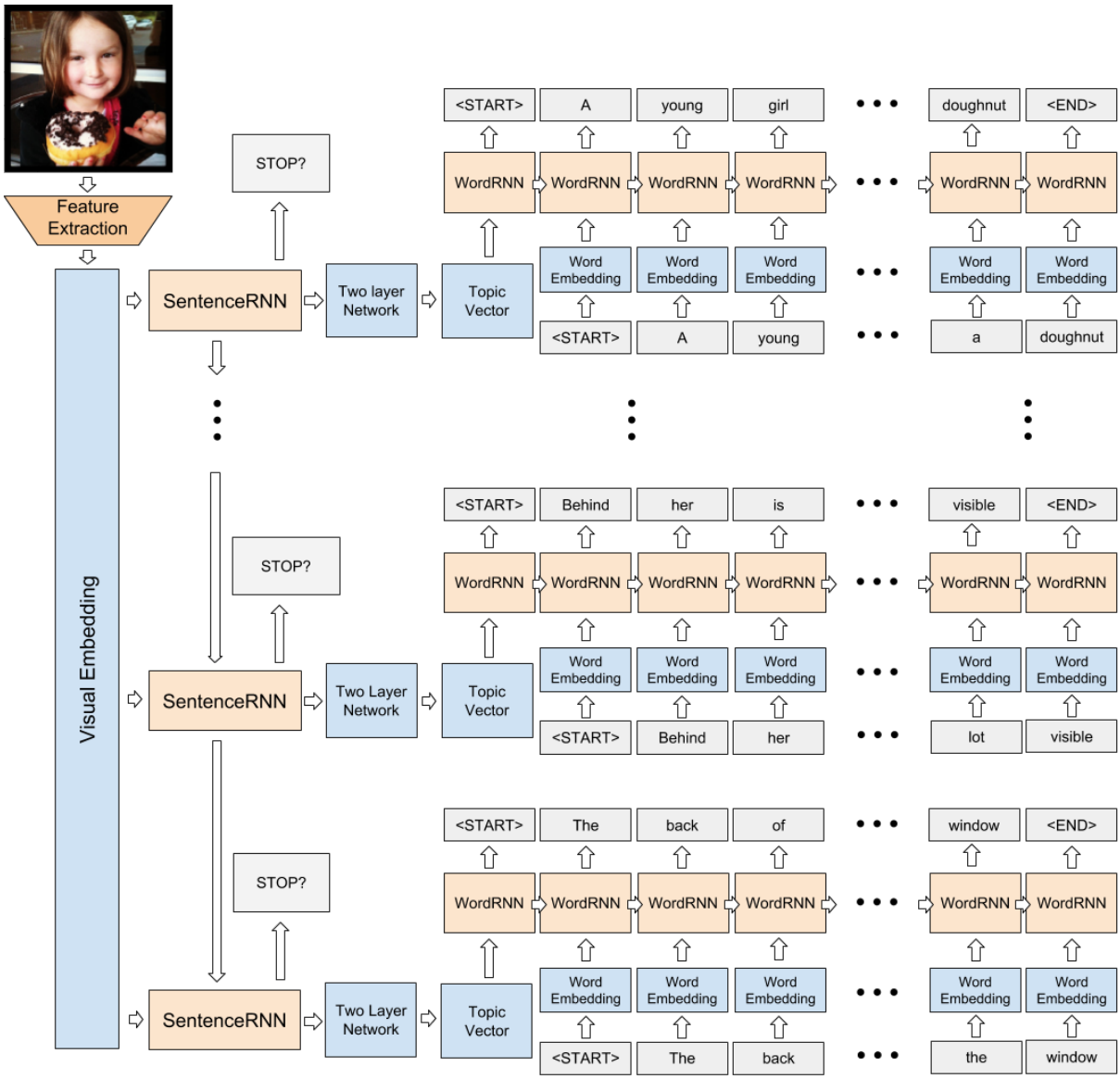


Figure 3.5: Hierarchical decoder.

In the hierarchical model the image features are not used to generate an image caption directly. Instead, first the SentenceRNN is run for M_{MAX} time-steps. The SentenceRNN outputs from each timestep are used for two purposes:

1. Output is passed through the stopping classifier, implemented as a pair of logistic units, learning the probability of the current sentence being the last one in the paragraph. We implement this unit by outputting a pair of values, unlike the commonly used approach of using a neural network layer with a single output for performing binary classification.

   The reason for this choice of implementation is that during the experiments, we discovered that a two-unit neural network provides slightly better results as measured in standard metrics, compared to using a single output. It could be that a network with two-unit output learns a slightly better representation for the stopping classifier. At the same time it is possible that this may cause the model to overfit more easily, and make it slower to converge;

2. The SentenceRNN outputs are fed into a two-layer fully connected network to generate E-dimensional topic vectors $\boldsymbol{T_i}$. We use ReLU [22, 45] non-linearity between the layers. The topic vectors $\boldsymbol{T_i}$ are each fed to the WordRNN, which generates each sentence.

The implementation of the baseline hierarchical model in PyTorch is reasonably straightforward. We use the same mini-batch size of 128 as for training the flat model. When training the model, the bigger challenge is ensuring that the PackedSequence object representing all sentences at position i in the mini-batch is sorted correctly. In general, for each mini-batch, the paragraph data must be sorted in M_{max} ways. To ensure that for each sentence at each SentenceRNN time-step the sorting order is correct, we need to maintain a separate sorting index for each sentence in the paragraph, sorting and unsorting them as required. A large portion of testing effort was dedicated to making sure that the above is working as specified.

3.3 Paragraph Captioning Pipeline

Combining the DenseCap-based encoder with the hierarchical decoder network results in the full "Regions-Hierarchical" model illustrated in Figure 3.6. The output of the dense captioning encoder is pooled into a single vector, which then serves as a context to the hierarchical RNN that generates the sentences.

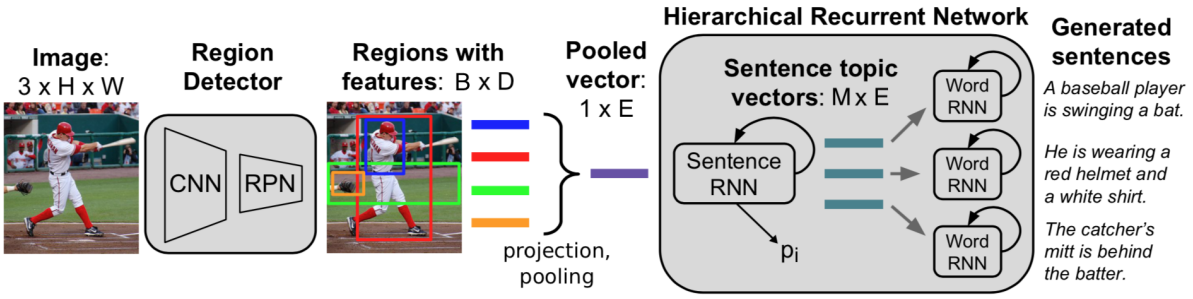


Figure 3.6: Encoder-Decoder architecture for paragraph captioning, adapted from Krause et al. [30], with variable names altered for consistency.

Let each training input be defined as a pair $(\boldsymbol{I}, \boldsymbol{S_P})$, where $\boldsymbol{I}$ is an image, and $\boldsymbol{S_P}$ is a paragraph caption for the image. Paragraph $\boldsymbol{S_P}$ contains M sentences. Sentence i has N_i words. Also, let $\boldsymbol{S_{ij}}$ be the jth word in the ith sentence. The model is trained to learn two probability distributions:

- p_i – a probability over $\{STOP, CONTINUE\}$ of the current sentence being the last sentence in the paragraph, so that

$$p_i(STOP) + p_i(CONTINUE) = 1, \tag{3.1}$$

- p_{ij} – a probability over the entire vocabulary for a given word being in position j of the ith sentence.

The loss proposed by Krause et al. [30] and subsequently adapted by others [9, 35] is composed of two components, one for each level of the decoder's hierarchy. These two terms of the loss function, $\mathcal{L}$, are the weighted *sentence loss* ℓ_{sent} and the weighted *word loss* ℓ_{word}, where ℓ_{sent} is the objective function for the SentenceRNN and ℓ_{word}, the objective function for the WordRNN:

$$\mathcal{L}(\boldsymbol{I}, \boldsymbol{S_P}) = \lambda_{sent} \sum_{i=1}^{M} \ell_{sent}(p_i, \mathbb{I}[i = M]) \; + \lambda_{word} \sum_{i=i}^{M} \sum_{j=1}^{N_i} \ell_{word}(p_{ij}, \boldsymbol{S_{ij}}). \tag{3.2}$$

The sentence loss, ℓ_{sent} is the cross-entropy loss on the "stopping" distribution p_i, where $\mathbb{I}[\kappa]$ is an indicator function that outputs 1 if the condition κ is satisfied, and 0 otherwise. The word loss, ℓ_{word} is the cross-entropy loss on word distribution p_{ij}. Finally, λ_{sent} and λ_{word} are the weighing terms. Krause et al. [30] set the weights to $\lambda_{sent} = 5.0$ and $\lambda_{word} = 1.0$, and we use the same values in our experiments. Assuming that each paragraph typically has 5-6 sentences, this weighing of loss terms makes the sentence loss and the word loss contribute to the final loss more equally.

3.4 Hierarchical-Coherent Model

The small number of training examples in the Stanford-Paragraph dataset may not contain all the possible linguistic patterns that are required for the generated paragraph-caption to feel as if it could have been produced by a human. It can also be hard for the hierarchical model to generate paragraphs where the language flows naturally from one sentence to the next [9, 35]. There are a few qualities of the captions that make them more "human-like", among them – how *diverse* they are – meaning do they follow the same template or do they seem to be more free-form, and how *natural* they are – how plausible it is that a human would have come up with a similar description. In this section, we will explore one potential approach for addressing the perceived quality of captions in more detail, while the next section will briefly cover several other potential improvements.

The hierarchical approach models multiple levels of structure for each paragraph. The baseline model, however, does not explicitly link the previous sentence to the next. Once the SentenceRNN in the baseline hierarchical model has produced the topic vector, it is used directly to generate the next sentence. These topic vectors do not depend on the output of the WordRNN, as they are produced before the WordRNN has been executed.

Our *hierarchical-coherent* model is based on the "Diverse-Coherent" model by Chatterjee and Schwing [9], in which they have extended the earlier, "Regions-Hierarchical" model with an aim to make paragraph captions appear more human-like. The high-level overview of the model is shown in Figure 3.7. Similarly to Krause et al. [30], the SentenceRNN in their model is used to generate sentence topic vectors $\boldsymbol{T_i}$ for each sentence in the

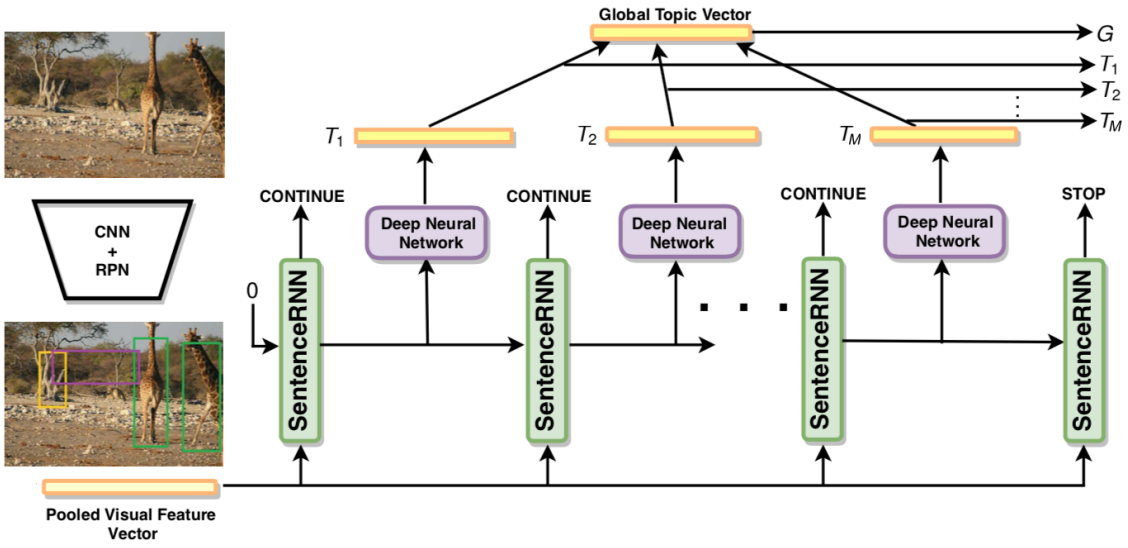


Figure 3.7: Hierarchical-coherent model overview – sentence topic and global topic creation, adapted from Chatterjee and Schwing [9], with variable names altered for consistency.

paragraph. However, unlike in the previous work, once all of the sentence topics are generated they are further processed to add coherence between sentences. Note, that the published "Diverse-Coherent" model also implements an optional variational auto-encoder component which aims to add diversity to the paragraph captions, but we did not implement this capability in our hierarchical-coherent model.

We will now describe how the model by Chatterjee and Schwing [9] establishes coherence between the individual sentences. First, once the topic vectors have been generated, their weighted sum is stored in a global topic vector $\boldsymbol{G}$:

$$\boldsymbol{G} = \sum_{i=1}^{M} \alpha_i \boldsymbol{T_i} \qquad \text{, where } \alpha_i = \frac{\|\boldsymbol{T_i}\|_2}{\sum_j \|\boldsymbol{T_j}\|_2}. \tag{3.3}$$

The *coherence vector* $\boldsymbol{C_{i-1}}$ is computed to help maintain coherence between the sentences. In order to compute the coherence vector, the hidden state of the WordRNN is retrieved after the last word of the current sentence has been generated. This hidden state representation is passed through a two-layer fully connected net, which the authors refer to as a *coherence network*. $\boldsymbol{C_{i-1}}$ is the output of this network. The initial coherence vector $\boldsymbol{C_0}$ is set to zero.

We now have the global topic vector $\boldsymbol{G}$, topic vector $\boldsymbol{T_i}$, and the coherence vector $\boldsymbol{C_{i-1}}$. Armed with these three vectors the final topic vector $\boldsymbol{T_i'}$ can be calculated with the aid of a *coupling unit*. Note, that the index $i-1$ of the coherence vector refers to the previous sentence, while the index i of the topic vector and final topic vector refer to the sentence about to be generated.

The coupling unit illustrated in Figure 3.8 is simply a sub-network that combines $\boldsymbol{G}$, $\boldsymbol{T_i}$ and $\boldsymbol{C_{i-1}}$ into a single output – a final topic vector $\boldsymbol{T_i'}$. The coupling unit is composed of a *fusion unit* and a *gating unit*. The fusion unit calculates $\boldsymbol{T_i^C}$, which is a weighted sum of $\boldsymbol{C_{i-1}}$ and $\boldsymbol{T_i}$:

$$\boldsymbol{T_i^C} = \frac{\alpha \boldsymbol{T_i} + \beta \boldsymbol{C_{i-1}}}{\alpha + \beta}. \tag{3.4}$$

This formulation itself comes from minimizing the squared norm of the difference between the fused topic-vector and each of its inputs:

$$\boldsymbol{T_i^C} = \arg\min_{\hat{\boldsymbol{T}}_i^{\boldsymbol{C}}} \alpha \left\| \boldsymbol{T_i} - \hat{\boldsymbol{T}}_i^{\boldsymbol{C}} \right\|_2^2 + \beta \left\| \boldsymbol{C_{i-1}} - \hat{\boldsymbol{T}}_i^{\boldsymbol{C}} \right\|_2^2 \quad \alpha, \beta \geq 0. \tag{3.5}$$

Here, α and β are constants set to different values depending on which dataset the model is trained on. For training on Stanford-Paragraph dataset, the values are $\alpha = 1.0$ and $\beta = 1.5$. Intuitively, the fusion unit can be thought of as producing a weighted combination of the coherence vector and the topic vector, with weighing being adjustable by changing the hyperparameters α and β.

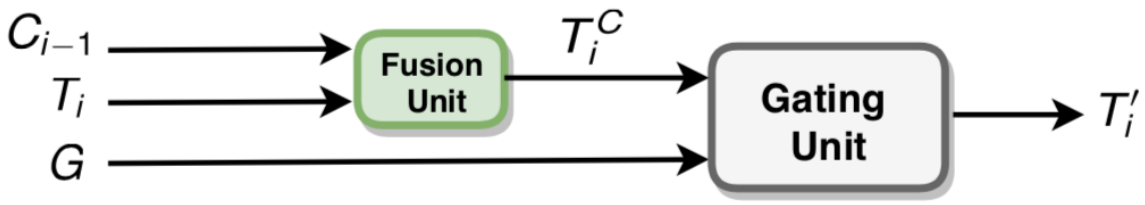


Figure 3.8: Coupling unit combines global topic vector $\boldsymbol{G}$, current topic vector $\boldsymbol{T_i}$ and the coherence vector $\boldsymbol{C_{i-1}}$ emmanating from the previous sentence into a final topic vector $\boldsymbol{T_i'}$ [9].

Next, the *fused* vector is passed to the *gating unit*, implemented as a single GRU cell, where the initial hidden state of the cell is initialized from the global topic vector $\boldsymbol{G}$. The cell is run for a single iteration, and its output is then the final topic vector $\boldsymbol{T_i'}$ that is used as the first input to the WordRNN generating the next sentence. This process is illustrated in Figure 3.9.

The authors of the model have shown that the coupling from the last generated word of the previous sentence to the initial input of the next sentence improves the overall coherence between the generated sentences. In a similar way, Li et al. [34] also consider tracking a global, paragraph level context akin to vector $\boldsymbol{G}$. However, they chose to use a third level of RNN hierarchy for modeling paragraph level context. The way in which the GRU cell is used in the coupling unit is in fact similar to this earlier work, now implemented in a more light-weight fashion.

The gating unit learns to control how much of the global context $\boldsymbol{G}$ to "let through" for generating the next sentence, instead of relying on the intermediate context vector $\boldsymbol{T_i^C}$ more local to the current sentence. The GRU cell in the gating unit is not modeling a recurrence because it is run for a single time-step only.

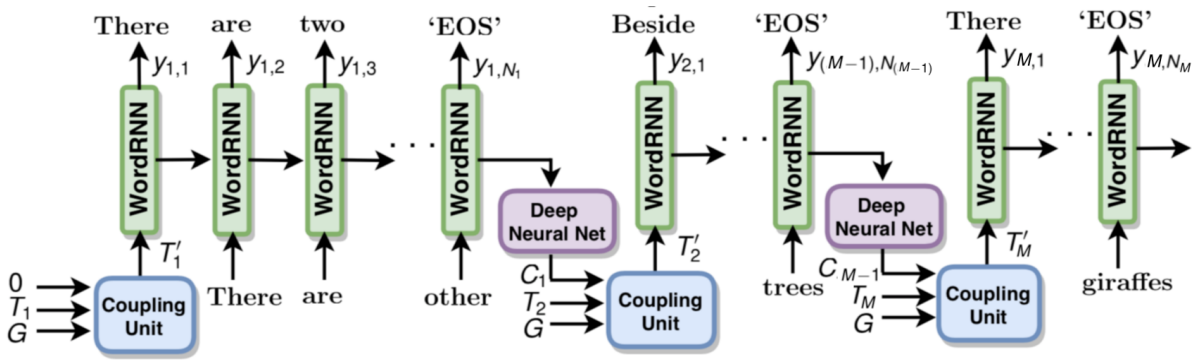


Figure 3.9: WordRNN receives topics from the coupling unit, figure adapted from Chatterjee and Schwing [9], with variable names altered for consistency.

3.4.1 Hierarchical-Coherent Decoder Implementation

The hierarchical-coherent model is implemented as a modification to a basic hierarchical model. The changes that were added to our hierarchical model are quite simple, with the exception of having more elements to be sorted at each sentence time-step i, to accommodate for PyTorch's PackedSequence-based RNN input format covered in Sections 2.3.3 and 3.2.1. The vectors $\boldsymbol{T_i}, \boldsymbol{G}, \boldsymbol{C_{i-1}}, \hat{\boldsymbol{T}}_{\boldsymbol{i}}^{\boldsymbol{C}}$, and $\boldsymbol{T_i'}$ are all E-dimensional, matching the dimension of input embeddings provided to the model at each WordRNN time-step.

To obtain the coherence vector $\boldsymbol{C_{i-1}}$ we use the second last hidden output of the WordRNN at sentence $i-1$. Our approach differs from the implementation seen in the partial source code examples* provided by the original authors of the architecture. The model authors use the hidden state of the WordRNN from the final time-step *after* the sentence has finished generating and the *<END>* token has been output. We tried doing the same in our experiments, but got consistently lower performance in terms of standard metrics.

The original authors [9] use the recently proposed Scaled exponential Linear Unit (SeLU) [29] activation function, which has a shape similar to the commonly used ReLU. However, we decided to use the ReLU non-linearity for both the topic generation network and the coherence vector generation network, since our experiments with SeLU did not yield comparatively better results.

3.5 Proposed Extensions to Paragraph Captioning

Several improvements to the baseline "Regions-Hierarchical" model by Krause et al. [30] have been published. Apart from the already covered introduction of coherence vectors and accompanying sentence-to-sentence coupling, these improvements include:

- Attention mechanisms on both the language and visual features [35],
- Adversarial objective function using a GAN architecture [13, 35],
- Variational auto-encoder based Encoder-Decoder architecture [9],
- Using a paragraph-level RNN (alongside Sentence- and WordRNNs) for tracking the global context [35].

The use of attention in hierarchical language models has been proposed by Li et al. [34]. Their attention mechanism was inspired by the use of attention in both the image captioning [71] and natural language translation [5] domains. Yu et al. [73] also propose the

*https://github.com/metro-smiles/CapG_RevG_Code

use of attention over visual features obtained inside a video frame in the closely related task of paragraph-length video captioning. In addition they employ temporal attention over video frames, which is less relevant to the task of creating captions for static images.

Liang et al. [35] propose to add several improvements for generating paragraphs, in their model called "Recurrent Topic-Transition GAN", also known as "RTT-GAN". The underlying generating model is similar to Krause et al. [30]. Their model also uses DenseCap features as input, and in their implementation the attention is both over the image-region vectors, as well as words in the corresponding dense captions generated by the same DenseCap network as image features. In addition, they incorporate a copying [21] mechanism to copy words from the attended region descriptions. Dai et al. [13] introduce a hierarchical evaluator in their GAN-based approach to paragraph captioning.

The previously mentioned "RTT-GAN" is also augmented with an adversarial training mechanism. Semi-supervised elements are added for training on paragraphs for which the images are not available, coupled with learning single-sentence captions from the MS-COCO Captions dataset. The model has two objectives: adversarial, where discriminators are optimized jointly with the generator, and a reconstruction loss of generated paragraphs.

As an alternative to using an adversarial objective, Chatterjee and Schwing [9] show that the diversity of generated captions can be improved by employing a VaE-based formulation, where part of the input space for the captioning model is sampled at run-time, thus essentially introducing some degree of randomness into the output of the model.

4. Experiments

We performed experiments on three different models – the flat model originally developed for image captioning, described in Section 2 and the two hierarchical model types covered in Section 3. All experiments were implemented using an in-house framework.

Our models relied on external, pre-computed image features extracted from models pre-trained on two different visual recognition tasks – dense captioning and image classification. All our paragraph captioning models used network weights of an RNN pre-trained on an image captioning task.

In Section 4.1 we describe the datasets used. Section 4.2 describes high-level evaluation criteria which can be used by humans and provides details on the popular automatic evaluation metrics we used for evaluating our results. Training details are described in Section 4.3. Finally, the results of our experiments are presented in Section 4.4, and the concluding remarks and discussion in Section 4.5.

4.1 Datasets

Three different datasets were used for the experiments. Before training our models, we divide each dataset into a *training set*, containing the images used for training our models and a *validation set*, which is a hold-out set that we use for verifying how well our models generalize to the previously unseen data.

MS COCO Captions [10] is an image captioning dataset which contains pairs of images and their corresponding captions. Each caption is comprised of a single sentence, of an average length of 11.30 words [30]. In our experiments we follow the training/validation split defined in the 2014 release of the MS COCO Object Detection dataset [37], commonly referred to as "MS COCO 2014". In this train/val split there are $82,783$ images in the training set, and $40,504$ images in the validation set. We use the *c5* subset of MS COCO Captions, providing each image with on average 5 different alternative descriptions.

The Visual Genome Regions (VG Regions) [31] dataset contains a total of $108,077$ images.

Table 4.1: Number of training and validation examples used and vocabulary size for each dataset. Each image in the MS COCO Captions has approx. five captions. Each image in the VG Regions has on average 50 regions.

Dataset	**Training** (images / captions)	**Validation** (images / captions)	**Training vocab. size**
MS COCO Captions	82,783 / 414,113	40,504 / 202,654	9,957
VG Regions	77,398 / 3,684,063	5,000 / 237,362	19,804
Stanford-Paragraph	14,579	2,490	4,600

Every image in VG Regions has on average 50 per-region annotations, each containing a short region description and the coordinates for a bounding box outlining the described region. The average length of each region description is 5 words. For training purposes we use the same train/val split as used by the official DenseCap implementation*. The training set we used contains $77,398$ and validation set $5,000$ images.

The Stanford-Paragraph dataset [30] is a subset of $19,551$ VG Regions images, each annotated with multi-sentence descriptions, each typically containing between 5 and 6 sentences, where each sentence is on average 11.91 words long. The descriptions are created from processed VG Regions descriptions, and they sometimes are similar to concatenations of region captions. However, Stanford-Paragraph is currently the best available paragraph captioning dataset and we use it for training our final models. We used the same train/val split as the authors of the dataset, with training on $14,579$ and validating on $2,490$ images.

Each dataset contains a different total number of words in its captions. We chose the words for each dataset-specific vocabulary by counting the occurrences of each word present in the captions, and adding words occurring four or more times to the corresponding vocabularies.

Table 4.1 shows dataset statistics. Note, that even though MS COCO Captions training set contains only $82,783$ unique images, the availability of around five different captions per each image effectively extend the size of the dataset to a much larger $414,113$ training examples.

All of the above datasets use images uploaded to Flickr by users, and each of them is a subset of the larger MS COCO Object Detection dataset [37]. Since we depend on transfer learning from models trained on larger single and region caption datasets to the models fine-tuned on smaller paragraph caption datasets, it is important to ensure that the final validation set used for paragraph captions does not overlap with the training sets used for

*https://github.com/jcjohnson/densecap/blob/master/info/densecap_splits.json

Table 4.2: Number of overlapping images between different training and validation sets.

Training dataset	Validation dataset	Overlap
MS COCO Captions / Train	Stanford-Paragraph / Val	5
Visual Genome Regions / Train	Stanford-Paragraph / Val	0
Visual Genome Regions / Train	MS COCO Captions / Val	87

pre-trained models. This ensures that the final models generalize well to unseen images.

As can be seen in Table 4.2, transfer learning from MS COCO Captions and VG Regions to Stanford-Paragraph has practically no overlap. There is a small overlap in the scenario where an image captioning model to be trained and validated on MS COCO Captions is first pre-trained using Visual Genome Regions. However, this case does not affect our results, as the primary scope of this work has to do with paragraph captioning.

4.2 Evaluation Metrics

The output of the image captioning task is similar to machine translation: either an image or text in a different language is mapped to a sequence of words. Several high level criteria for human evaluation of machine translation exist, some of them proposed as early as 1966 [46]. Among them are: fidelity, intelligibility [46, p. 68], adequacy, and fluency [69]. These criteria can then be applied to the text produced by an image captioning or machine translation model on a sentence by sentence basis. Short definitions for each of these criteria are as follows:

- *Fidelity* – For machine translation fidelity measures how well the translated sentence conveys the information content of the source sentence. In the case of captioning we look at the input image, and try to assess if the generated caption conveys the information that we feel is relevant for a given length description. What is happening in the image on the high level? What is important in the image? What can be ignored? Are the details reproduced correctly – or is the caption making mistakes in colors, numbers, types of entities?

- *Intelligibility* – Originally defined to measure how intelligible the translation is without comparing it with the source sentence. In the case of captioning, we disregard the input image and evaluate if we can understand the resulting caption given our knowledge of the surrounding world. In particular: Is the grammar something that can occur in a real English sentence? Is the phrase being used meaningful?

- *Adequacy* – In machine translation it measures how well the translated sentence conveys the information content of the control translation. When evaluating image captions we ignore the input image and look at ground truth captions. How similar is our generated description to the ground truth caption? Does the generated caption cover the same points, or are there big differences?

- *Fluency* – Measure whether the translation reads like good English, without reference to the original content or control translations. When applying to the task of image captioning, we ignore both the input image and the ground truth caption, and try to assess if the caption reads like good English. Note, that this is not the same as intelligibility, as we may have an intelligible caption, which reads like broken English.

To see how fast we can apply the above metrics, and to see how well these criteria, originally devised for automatic translation evaluation, fit in the context of paragraph captioning we sampled a set of six images from our results for several of our models, and applied these criteria using scores of 1 (low), 2 (medium), 3 (high), with results shown in the Appendix. Evaluating captions for six images produced by seven models – or 42 captions in total took us close to two hours. Clearly, evaluating tens of models on a validation set containing 5000 (image, caption) pairs would require a large pool of workers, and may still take months to complete.

Even though we only chose a small sample of six images for evaluation, we saw a large variety in the quality of the captions generated by each model. In addition, the caption scores for the four human evaluation criteria appeared to be similar. For example, the generated captions that scored "high" on fidelity tended to score "medium" or "high" on the other three criteria as well.

Relying on human judgment is clearly not practical when evaluating results of large datasets, and for this reason, automatic measures are used. The aim of the automatic metrics is to serve as a proxy to human evaluation [48], and several such metrics have been proposed. These metrics require access to ground truth captions, and they can directly measure only the adequacy of the generated captions. In the next subsection we cover the metrics that we used to evaluate the paragraph captions generated by our models.

4.2.1 Validation Loss

We train our models by minimizing the loss function, defined in Section 2.4 for the flat model, and in Section 3.3 for the two hierarchical models. Therefore, the basic automatic metric that can be used to assess the quality of the generated captions is the *validation*

loss, defined as the average of the loss function output computed on the captions generated for the images in the hold-out validation set.

Validation loss can be calculated quickly at training time, and therefore can be used as a simple indicator of how well the model generalizes at any point during training. However, as discussed in Sections 4.4.1 and 4.4.3, it is a poor indicator of the quality of the paragraph captions, when compared to other automatic metrics.

4.2.2 Precision and Recall

The concepts of *precision* and *recall* [52] are useful for understanding the automatic measures used for caption evaluation.

The precision, P, is defined as:

$$P = \frac{TP}{FP + TP}, \tag{4.1}$$

where TP is the number of true positives – in case of image captions this refers to the number of words that occur in both the ground truth and the generated caption, and FP the number of false positives – the number of words that occur in the model output, but not in the ground truth.

The recall, R, is defined as:

$$R = \frac{TP}{TP + FN}, \tag{4.2}$$

where FN is the number of false negatives, or words that are present in the ground truth but absent from the generated caption.

4.2.3 BLEU

BLEU-{1,2,3,4} (bilingual evaluation understudy) [48] is a machine translation evaluation metric based on precision over n-grams, where an n-gram is a sequence of consecutive words of length n. For example, in the caption "the cat is on the mat" we would have:

1-grams (or unigrams):	"the", "cat", "is", "on", "the", "mat"
2-grams (or bigrams):	"the cat", "cat is", "is on", "on the", "the mat"
4-grams:	"the cat is on", "is on the mat"

Note, that a related concept of a "character n-gram" exists, where n refers to the number of individual characters instead of words. In our case, we are only interested in word n-grams.

In the following examples we will illustrate BLEU-1 which is applied to unigrams, however the same approach generalizes to n-grams of any size. Let Count(u) be the number of words (or unigrams) in the caption that are also present in the ground truth, with each occurrence of a word increasing the count. We apply a unigram precision score to every caption we generate in the following manner:

$$P_{\text{caption}} = \frac{\sum_{u \in \text{Caption}} \text{Count}(u)}{L}. \tag{4.3}$$

Given a generated caption of length L, we increase the true-positive count every time we encounter a word u that also occurs in the ground truth. The problem with simple precision is that the true-positive counts may exceed the number of occurrence of the word in the ground truth. To illustrate this, we provide an example, adapted from Papineni et al. [48].

Ground truth: the cat is on the mat.
Caption 1: the the the the the the the.
Caption 2: There is a cat on the mat.

For Caption 1, the standard unigram precision score would be 7/7, and for Caption 2 it would be 3/7, even though from the human perspective the 2nd caption is much closer to the ground truth.

BLEU seeks to remedy this problem by employing *modified precision*. The modified precision for unigrams for a single caption, P_1^*, is calculated by simply clipping the count of occurrences of each word in the generated caption to the maximum number of occurrences in the ground truth:

$$P_1^* = \frac{\sum_{u \in \text{Caption}} \text{Count}_{\text{clip}}(u)}{L}. \tag{4.4}$$

Based on this equation, the modified unigram precision score for Caption 1 becomes 2/7 and for Caption 2 is again 3/7, which is now higher than for the obviously wrong Caption 1. In general, the higher the precision score, the higher is then the resulting BLEU-N score.

The general form of a modified n-gram precision, P_n, is applied to the entire set of candidate captions:

$$P_n = \frac{\sum_{c_i \in \text{candidates}} \sum_{n\text{-gram} \in c_i} \text{Count}_{\text{clip}}(n\text{-gram})}{\sum_{c_i \in \text{candidates}} \sum_{n\text{-gram} \in c_i} \text{Count}(n\text{-gram})}. \tag{4.5}$$

The modified precision on its own favors shorter captions, because it leaves less room for the model to make mistakes. In order to combat this, and reward longer captions, BLEU introduces a *brevity penalty*, BP:

$$\text{BP} = \begin{cases} 1 & \text{if } \mathcal{C} > \mathcal{S} \\ \exp\left(1 - \frac{\mathcal{S}}{\mathcal{C}}\right) & \text{if } \mathcal{C} \leq \mathcal{S} \end{cases}. \tag{4.6}$$

Here, $\mathcal{C} = \sum |c_i|$ is the sum of lengths of all the candidate captions, and $\mathcal{S} = \sum |s_i|$ the sum of lengths of all the corresponding ground truth captions.

The final BLEU-N score is calculated by multiplying the weighted geometric mean of modified n-gram precision scores, P_n, for $n = 1, \ldots, N$ by a brevity penalty:

$$\text{BLEU-N} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \ln P_n\right). \tag{4.7}$$

Here, the weights are typically set so that $w_n = 1/N$. Intuitively, BLEU-2 combines the modified precision P_1 and P_2, BLEU-3 combines P_1, P_2, and P_3, etc.

Returning to our earlier example, where we have a single pair of captions being compared, we can see that the generated caption "There is a cat on the mat." is longer than the ground truth caption, so we do not incur any brevity penalty. In addition, the geometric mean of a single number is the number itself, so our final BLEU-1 score is $3/7 = 0.4285$.

The raw output of the BLEU-{1,2,3,4} metric is in the range $(0, 1)$, however, this output is usually multiplied by a factor of 100, to map the final score into a $(0, 100)$ range. We follow the same approach when reporting our results.

4.2.4 Meteor

There are several issues when using BLEU as an automatic image caption evaluation metric. First of all, it doesn't account for recall, defined in Section 4.2.2. When applied to the task of image captioning, recall measures how many of the words in the ground truth occur in the generated caption. In addition, BLEU doesn't account for synonyms, and in the case of larger n-grams, for example when calculating BLEU-4, it penalizes the captions that have a different, yet correct, word order when compared to the ground truth.

Meteor [6, 14] is a metric that combines both precision and recall and seeks to remedy the problems of matching synonyms and paraphrased sentences. In this section we will refer to the generated sentences (or captions) as the "hypothesis" and the ground truth

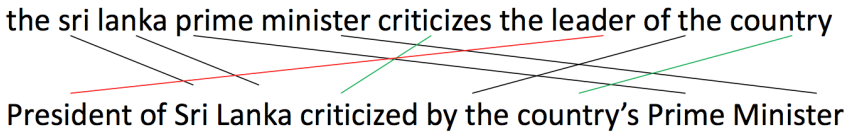


Figure 4.1: Few examples of Meteor matches [33].

sentences (or captions) as the "reference", to use the same terms as used by the authors of Meteor.

Meteor tries to match reference sentences to hypothesis sentences using four types of matchers, m_i:

- **Exact**, m_1 – matching identical word forms, similar to unigram counts used in BLEU;
- **Stem**, m_2 – matching words that share a stem;
- **Synonym**, m_3 – matching words occurring within the same WordNet [43] synset, used as a set of synonyms corresponding to each word;
- **Paraphrase**, m_4 – matching phrases according to a predefined paraphrase table.

Figure 4.1 shows examples of matches between sentences. The black lines correspond to "exact" matches, green lines show "stem" matches, and the red line shows an example of a "paraphrase" match.

Meteor treats the words above a certain frequency threshold in a given language as the *function words*. Examples of function words are "a", "the", "have", "about', "people", "could". The words that are less frequent are considered *content words* – such as "computer", "dog", "purple", "play", etc. The function words in hypothesis and reference sets are denoted as h_f and r_f, and content words as h_c and r_c respectively.

Each matcher m_i has an associated weight w_i (see Table 4.3 for parameter values). The relative weighing between content and function words is determined by the value of the parameter δ. The matchers and the corresponding weights are used to calculate the weighted precision P_w and weighted recall R_w:

$$P_w = \frac{\sum_{i=1}^{4} w_i \cdot (\delta \cdot m_i(h_c) + (1-\delta) \cdot m_i(h_f))}{\delta \cdot |h_c| + (1-\delta) \cdot |h_f|}, \tag{4.8}$$

$$R_w = \frac{\sum_{i=1}^{4} w_i \cdot (\delta \cdot m_i(r_c) + (1-\delta) \cdot m_i(r_f))}{\delta \cdot |r_c| + (1-\delta) \cdot |r_f|}, \tag{4.9}$$

$$\tag{4.10}$$

where $|h_c|$, $|h_f|$, $|r_c|$, and $|r_f|$ are the numbers of content and function words in the hypothesis and reference captions respectively.

Next, the parameterized harmonic mean F_{mean} of the weighted precision and recall is calculated:

$$F_{\text{mean}} = \frac{P_w \cdot R_w}{\alpha \cdot P_w + (1-\alpha) \cdot R_w}, \tag{4.11}$$

where parameter α (see Table 4.3) is usually chosen in such a way as to weigh recall higher than precision [65]. In order to reward longer consecutive matches, Meteor introduces "chunks" defined as sequences of matches that are in the same order in both the hypothesis and the reference. For example, in Figure 4.1 "sri lanka" and "prime minister" each represent a chunk composed of two matches. The longer the chunks, the better the Meteor score is, because then the word order between the reference and the hypothesis is more similar, than if the number of chunks is large, in which case the hypothesis is more "fragmented".

Let m be the total number of word matches (averaged over the hypothesis and the reference, to take paraphrasing into account), and let k be the total number of chunks, then the fragmentation penalty is calculated as follows:

$$\text{Penalty} = \gamma \cdot \left(\frac{k}{m}\right)^{\beta}. \tag{4.12}$$

The final Meteor score is then calculated by applying the fragmentation penalty to the harmonic mean of the weighted precision and recall:

$$\text{Score} = (1 - \text{Penalty}) \cdot F_{\text{mean}}. \tag{4.13}$$

We use the default values for the English version of Meteor, shown in Table 4.3, tuned to increase the correlation between human judgment and Meteor-scores [14]. All in all, Meteor has been shown to correlate with human judgment better than BLEU [14].

To illustrate, we calculate the Meteor score for the example sentences, given in the previous section by using only the *exact* matches, m_1. Here, a period character (".") is also considered a function word. We are given the reference and the hypothesis captions, with content words underlined, and the remaining words considered to be function words:

Hypothesis: *There is a cat on the mat.*
Reference: *the cat is on the mat.*

Table 4.3: Meteor parameter values used in scoring the English-language sentences [14].

α	β	γ	δ	w_1	w_2	w_3	w_4
0.85	0.20	0.60	0.75	1.00	0.60	0.80	0.60

Based on the above, we have:

$$h_c = (\text{“cat”}, \text{“mat”}), \quad h_f = (\text{“there”}, \text{“is”}, \text{“a”}, \text{“on”}, \text{“the”}, \text{“.”})$$
$$r_c = (\text{“cat”}, \text{“mat”}), \quad r_f = (\text{“the”}, \text{“is”}, \text{“on”}, \text{“the”}, \text{“.”})$$
$$m_1(h_c) = m_1(r_c) = 2, \quad m_1(h_f) = m_1(r_f) = 4$$
$$|h_c| = 2, \quad |h_f| = 6, \quad |r_c| = 2, \quad |r_f| = 5.$$

Next, we use the Eqs. 4.8 and 4.9 to calculate the weighted precision and recall. In order to keep our example simple, we only calculate the *exact* matches, m_1:

$$P_w = \frac{w_1 \cdot (\delta \cdot m_1(h_c) + (1-\delta) \cdot m_1(h_f))}{\delta \cdot |h_c| + (1-\delta) \cdot |h_f|} = \frac{1.00 \cdot (0.75 \cdot 2 + (1-0.75) \cdot 4)}{0.75 \cdot 2 + (1-0.75) \cdot 6} = 0.8333$$
$$R_w = \frac{w_1 \cdot (\delta \cdot m_1(r_c) + (1-\delta) \cdot m_1(r_f))}{\delta \cdot |r_c| + (1-\delta) \cdot |r_f|} = \frac{1.00 \cdot (0.75 \cdot 2 + (1-0.75) \cdot 4)}{0.75 \cdot 2 + (1-0.75) \cdot 5} = 0.9090,$$

and use the above to obtain the F_{mean}:

$$F_{\text{mean}} = \frac{0.8333 \cdot 0.9090}{0.85 \cdot 0.8333 + (1-0.85) \cdot 0.9090} = 0.8968.$$

The total number of matches in our example is: $m = m_1(h_c) + m_1(h_f) = 6$. We have $k = 3$ chunks: “*is*”, “*cat*”, and “*on the mat.*”, which we use to calculate the Penalty:

$$\text{Penalty} = \gamma \cdot \left(\frac{k}{m}\right)^{\beta} = 0.60 \cdot \left(\frac{3}{6}\right)^{0.20} = 0.5223.$$

The final Meteor score of 0.4284, for our example caption, is obtained by applying the Penalty term to the F_{mean}:

$$\text{Score} = (1 - \text{Penalty}) \cdot F_{\text{mean}} = (1 - 0.5223) \cdot 0.8968 = 0.4284.$$

The range of possible values for the final Meteor score is $(0, 1)$, similarly to BLEU-{1,2,3,4}. Typically the resulting score is multiplied by 100, which is what we do when displaying our results.

4.2.5 CIDEr

Another popular automatic evaluation metric is CIDEr (Consensus-based Image Description Evaluation) [65]. Unlike BLEU and Meteor, designed for evaluating the results of

machine translation, CIDEr has been developed with the explicit purpose of evaluating image captions. It supports datasets that have multiple ground truth image descriptions available for each image, such as MS COCO Captions [10]. In our paragraph captioning experiment, however, we will not use the support for evaluating multiple captions per image because the Stanford-Paragraph [30] validation dataset contains only one reference description per image.

In general, CIDEr may refer to either a regular CIDEr score, or to a modified version, known as CIDEr-D, which introduces modifications to the CIDEr score designed to prevent "gaming" the system by tailoring the captions to get artificially high scores. In our experiments we use the CIDEr-D version of CIDEr, following Karpathy et al. [26]. We will first introduce the original version of CIDEr, and then show how CIDEr-D differs from it.

To calculate the CIDEr score, the words in both the ground truth and the generated caption are converted to their stems. Next the Term Frequency Inverse Document Frequency (TF-IDF) [55] value is calculated for each n-gram. The TF-IDF value for each n-gram is based on the relative term frequency (TF) of the n-gram in the sentence (either ground truth or generated) multiplied by the Inverse Document Frequency (IDF) for the same n-gram. Thus, n-grams that are rare in the rest of the captions, but occur frequently for the current image, have a high TF-IDF value, and conversely the n-grams that occur frequently in different captions, will have low IDF, and therefore lower overall TF-IDF value.

Let s_i be the ground truth caption for the image i, and c_i be the caption generated by our model. Also, let $\boldsymbol{g^n}$ be the function that maps a caption to a TF-IDF value vector of size V^n where V is the size of dataset vocabulary (see Section 2.1.1), and n is the n-gram size. Every element in such a vector is a TF-IDF value for the corresponding n-gram in the caption, or is set to zero, if the n-gram is not in the caption. Vector $\boldsymbol{g^n}(c_i)$ thus represents the generated caption and vector $\boldsymbol{g^n}(s_i)$ represents the ground truth. Typically, four such vectors are formed with $n = \{1, 2, 3, 4\}$.

Normally, a CIDEr score takes into account multiple ground truth captions, however, the Stanford-Paragraph dataset contains only one ground truth caption per image. The n-gram CIDEr_n score for when there is only one ground truth sentence is the cosine similarity between the generated caption and the reference caption:

$$\text{CIDEr}_n(c_i, s_i) = \frac{\mathbf{g^n}(c_i) \cdot \mathbf{g^n}(s_i)}{\|\mathbf{g^n}(c_i)\| \, \|\mathbf{g^n}(s_i)\|}. \tag{4.14}$$

Next, the n-gram specific CIDEr_n scores are combined to form the final CIDEr score:

$$\text{CIDEr}(c_i, s_i) = \frac{1}{4}\sum_{n=1}^{4} \text{CIDEr}_n(c_i, s_i). \tag{4.15}$$

When more than one reference caption is available, the CIDEr_n score is calculated as an average of cosine similarities for each reference caption.

The cosine similarity score of TF-IDF vectors accounts for both precision and recall [65]. It has also been shown [65] that CIDEr and Meteor correlate the closest to human judgment, and therefore we use these as our main metrics.

CIDEr-D extends the CIDEr score by applying several modifications: the removal of stemming, adding of Gaussian penalty for the difference of length between the ground truth and the generated caption, and the clipping of n-gram counts in the generated caption to not exceed the corresponding n-gram counts in the ground truth (similarly to BLEU).

The n-gram specific CIDEr-D_n score is calculated as follows:

$$\text{CIDEr-D}_n(c_i, s_i) = 10 \cdot \exp\left(\frac{-(|c_i| - |s_i|)^2}{2\sigma^2}\right) \cdot \frac{\min(\mathbf{g^n}(c_i), \mathbf{g^n}(s_i)) \cdot \mathbf{g^n}(s_i)}{\|\mathbf{g^n}(c_i)\| \, \|\mathbf{g^n}(s_i)\|}. \tag{4.16}$$

Here, $|c_i|$ and $|s_i|$ are the numbers of words in the generated and ground truth captions, respectively. The value of the parameter σ is typically set to 6, which is what we use in our experiments. The score is multiplied by the constant 10 to make the metric numerically comparable to other metrics.

To illustrate the steps needed to compute the CIDEr and CIDEr-D scores we will show an example of calculating unigram CIDEr_1 and CIDEr-D_1 scores for the example hypothesis and reference captions already familiar to us from before. In order for the example to be meaningful, we also need to assume that there exists a large set of other hypotheses and reference captions, because it is not meaningful to calculate the IDF value for when there is only one caption in the set. To simplify things, we assume to have been given some TF-IDF values for each word in the hypothesis and the reference captions:

Hypothesis, c_i:	There	is	a	cat	on	the	mat	.
TF-IDF values:	0.	0.	0.	0.13	0.	0.	0.15	0.

Reference, s_i:	the	cat	is	on	the	mat	.
TF-IDF values:	0.	0.11	0.	0.	0.	0.19	0.

We set the values for frequently occuring words to 0., such that $0. = 0 + \epsilon$ is an arbitrarily small, non-zero number. The only values in our example significantly larger than zero are

"cat" and "mat", because in our example we assume that the other words occur frequently across different captions, making their IDF value approach zero. Words "cat" and "mat" occur exactly once in both the hypothesis and the reference caption, so we do not need to do any clipping.

Let us define a small vocabulary of unigrams of size $V = 11$, containing the tokens:

"a", "cat", "dog", "is", "mat", "on", "sit", "the", "there", "window", ".".

The unigram TF-IDF vectors, $\mathbf{g^1}(c_i)$ and $\mathbf{g^1}(c_i)$, are obtained by taking element-wise TF-IDF scores for each word in the vocabulary. The words, which are not in the corresponding caption have the associated values of exactly zero:

$$\begin{array}{lccccccccccc} & \text{a} & \text{cat} & \text{dog} & \text{is} & \text{mat} & \text{on} & \text{sit} & \text{the} & \text{there} & \text{window} & . & \\ \mathbf{g^1}(c_i) = \Big[& 0. & 0.13 & 0 & 0. & 0.15 & 0. & 0 & 0. & 0. & 0 & 0. & \Big]^T \\ \mathbf{g^1}(s_i) = \Big[& 0 & 0.11 & 0 & 0. & 0.19 & 0. & 0 & 0. & 0 & 0 & 0. & \Big]^T. \end{array}$$

The CIDEr_1 score is the cosine similarity of the above two vectors:

$$\text{CIDEr}_1(c_i, s_i) = \frac{\mathbf{g^1}(c_i) \cdot \mathbf{g^1}(s_i)}{\|\mathbf{g^1}(c_i)\| \, \|\mathbf{g^1}(s_i)\|} = 0.9821.$$

In our example, the number of occurences of the words "cat" and "mat" in the reference and the hypothesis captions are equal, so we do not need to do any clipping for the CIDEr-D score, and therefore can use the CIDEr_1 value we just obtained to calculate the corresponding CIDEr-D_1 score. We proceed by obtaining the Gaussian term that penalizes the difference in length.

We take the lengths of the hypothesis $|c_i| = 8$ and the reference $|s_i| = 7$ and calculate the Gaussian penalty term:

$$\exp\left(-\frac{(|c_i| - |s_i|)^2}{2\sigma^2}\right) = \exp\left(-\frac{(8-7)^2}{2 \cdot 6^2}\right) = \exp\left(-\frac{1}{72}\right) = 0.9862.$$

The unigram CIDEr-D score for our example is thus:

$$\text{CIDEr-D}_1 = 10 \cdot 0.9862 \cdot 0.9821 = 9.6854702.$$

The range of CIDEr-D scores for individual captions is $(0, 10)$. However, the final score is obtained by averaging individual caption scores, and when a large number of captions is scored, the upper range for the average CIDEr-D score over the entire validation set is much smaller, because many captions typically score very close to zero. Therefore, it is customary to multiply the final CIDEr-D score by 100, which is what we do when showing our scores.

4.3 Training Details

The experiments were performed using the DeepCaption [60] neural image captioning framework developed as part of the work done by the author of this thesis at the Content-Based Image and Information Retrieval group (CBIR)* at Aalto University. DeepCaption is implemented using the PyTorch open source deep learning platform and the Python programming language.

The highlights of the framework include:

- *Modular architecture* – support for adding multiple different neural network models, as well as combining and extending them;
- *Configuration management* – ability to define new datasets and external features, as well as combining these datasets and feature types and seamlessly train the models using the combined datasets;
- *Image captioning model* – robust baseline image captioning implementation similar to "Show and Tell" [66];
- *Hierarchical paragraph captioning model* – Combines two RNNs into a hierarchical structure for producing longer captions;
- *Transfer learning from image captioning to paragraph captioning* – Support for pre-loading pre-trained language model weights into the hierarchical model;
- *Logging and visualization* – Support for both simple logging, as well as recording training information and neural network state using TensorBoardX†.

Benefits of the PyTorch [49] platform include the ease of development and debugging. The platform doesn't require the developer to make a distinction between the two separate stages of building a static computation graph and on-demand code execution, as in TensorFlow [1]. This ability to modify computation graph at runtime is called *dynamic execution*. It allows for more granular control (such as arbitrary conditional statements) and facilitates the use of standard debugging techniques. These capabilities, in turn, aid in flexibility of development, increase the diversity of supported models and enable faster iteration in development and testing of neural network models.

In our experiments, all datasets were handled by a common data-loading mechanism built into the DeepCaption framework. As much as possible, dataset loading relied on the file

*https://research.cs.aalto.fi/cbir/
†https://github.com/lanpa/tensorboardX

folder structure as provided by the dataset creators, to ensure that the data-loading step is reproducible with minimum effort. This was further aided by the creation of a dataset-configuration file, where a DeepCaption user can define the paths to each of the datasets being used, making it easy to download and store (sometimes quite large) datasets at the location of choice.

4.3.1 Feature Extraction

The models trained all follow the basic Encoder-Decoder template, where incoming image features are encoded into an E-dimensional context vector, which is then fed into the decoder module, which implements the language model that generates the captions.

The encoder components of our models are responsible for providing the image features. We chose not to backpropagate the loss through the encoder for two reasons. First, this allows us to speed up the training of each epoch by using pre-computed image features obtained from two different pre-trained models – ResNet-152 [23] and DenseCap [25]. Furthermore, the implementation of the DenseCap network that we use to pre-compute the features is using an incompatible framework, making it impossible to backpropagate through it without full reimplementation in PyTorch. In addition, published research [9, 30, 35] on hierarchical paragraph captioning models uses DenseCap image features obtained from the same implementation. Unlike the other published models, we also use ResNet-152 features alongside the ones extracted from DenseCap, because they gave us better results on the image captioning task.

DeepCaption supports retrieving features from multiple different file formats, such as HDF5*, serialized NumPy† array, as well as LMDB‡. We choose LMDB for storing and retrieving image features, due to its high performance and lower required memory footprint. LMDB is implemented using low-level Linux memory-mapping, which means that its performance is highly optimized and very similar to fetching data from virtual memory.

The pre-trained ResNet-152 model comes bundled with the TorchVision set of libraries included with the PyTorch distribution. TorchVision makes it possible to extract features from several popular convolutional networks, including ResNet-152 inside DeepCaption, without relying on any code other than standard PyTorch libraries. We used this to evaluate different other alternatives such as VGG [59] and AlexNet [32].

To aid in this process, feature extraction scripts were developed to provide a bridge to the DenseCap implementation written in the Lua language. These scripts rely on the dataset

*https://www.hdfgroup.org/solutions/hdf5/
†http://www.numpy.org/
‡http://www.lmdb.tech/doc/

configuration management logic of DeepCaption to ensure that the filename-to-feature mapping remains consistent. Finally, the output of the feature extraction pipeline for DenseCap are the same kind of LMDB files as created for ResNet-152.

4.3.2 Hyperparameters

All models were trained using the Adam [27] learning rate (LR) optimizer. Adam provides a mechanism for adjusting individual per-parameter learning rates. However, we also need to set a starting LR for the whole model. For this, we experimented with three different learning rate schedules:

- *Constant* – setting a constant learning rate for the entire duration of training;
- *Reduce on Plateau* – scale learning rate down by some factor whenever a key loss metric – such as validation loss or scoring metric stops improving;
- *Cyclical* – continuously cycle the learning rate between two predefined values [61]. Depending on which variant is in use, the learning rate can increase and decrease linearly or for example quadratically in a sine-wave like fashion. An example of a cyclical learning rate with linear increase and decrease can be seen in Figure 4.2.

In all three cases the final learning rate is determined by the Adam optimizer, which takes the learning rate provided by the scheduler, be it static or changing, and then multiplies it with a per-parameter factor. For the cyclical learning rate, we used a triangular schedule, where the learning rate alternates linearly between the base and the maximum values once per epoch.

At inference, we cap the WordRNN output to 20 words per sentence, with an exception of the flat model used for paragraph captioning, where the corresponding length cap is

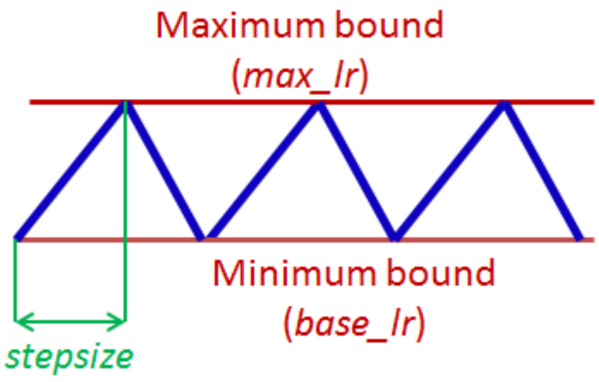


Figure 4.2: Triangular learning rate schedule [61].

set to 80 words. We set the maximum number of sentences our hierarchical models are allowed to generate to $M_{max} = 6$. All models are trained with mini-batches of size 128. Models are trained for the number of epochs it takes for the CIDEr and Meteor scores measured on the validation set to stop improving. In the case of image captioning, this may be closer to 20 epochs, while some hierarchical models were trained for as many as 250 epochs. In addition, all models are using the same seed for PyTorch's random number generator. This in turn means that running training with the same parameters provides identical results.

Our models were trained on either *Triton*[*] or *Taito*[†] scientific computing environments provided by Aalto University and CSC - IT Center for Science respectively. All our models were trained on either the nVidia K80 or P100 GPUs.

4.4 Results

In this section we discuss the results obtained for paragraph captioning using the three models: flat, hierarchical and hierarchical-coherent, which were introduced in Chapters 2 and 3. All three models depend on a pre-trained WordRNN trained on either the MS COCO Captions or VG Regions datasets. For the published baselines, we show the results reported by the authors of each baseline.

4.4.1 Image Captioning Experiments

We first trained a range of flat image captioning models using the MS COCO Captions dataset. In addition to pre-training a WordRNN to be later used on the paragraph captioning task, another important goal for this first set of experiments was to identify good ways for pre-computing input features extracted from the pre-trained ResNet-152 and DenseCap models, described in Sections 2.2.1 and 3.1.1. We also experimented with using average and maximum pooling, and different combinations of input features.

We noticed that our best-performing models on the image captioning task used average pooled DenseCap features. Interestingly enough, Krause et al. [30] specifically mention that their DenseCap features were maximum pooled. Intuitively, average pooling may in fact be better at captioning more general information about the image, by "averaging over" the small details. On the other hand, maximum pooling may help the model capture more of the specific detail, not all of which is relevant for simple single-sentence captioning. To

[*]https://scicomp.aalto.fi/

[†]https://research.csc.fi/csc-s-servers#taito

Table 4.4: Image captioning experiments – comparison with state of the art on the MS COCO Captions dataset. Our models used average-pooled DenseCap features. The scores in the top section of the table are taken from the published papers, the scores for our models are shown in the bottom section.

Model name	**CIDEr***	**Meteor**
Show and Tell [66]	85.5	23.7
Show, Attend and Tell [71]	-	23.9
Adaptive Attention [38]	108.5	26.6
Att2all [54]	114.0	26.7
Bottom-Up and Top-Down Attention [3]	**117.9**	**27.6**
Image Captioning / Flat, ResNet only, $E = 256$ (Ours)	87.2	24.3
Image Captioning / Flat, DenseCap only, $E = 256$ (Ours)	89.8	24.8
Image Captioning / Flat, DenseCap+ResNet, $E = 256$ (Ours)	93.3	25.2
Image Captioning / Flat, DenseCap+ResNet, $E = 1024$ (Ours)	93.2	25.3

address this question, we used the models trained on average as well as maximum pooled features for transfer-learning.

Table 4.4 shows the recent state of the art results in image captioning using the MS COCO Captions dataset. Our model is based on "Show and Tell" [66], but we score higher compared to this earlier model most likely because we use better input features. The other baseline models shown in the table use image captioning improvements which we briefly covered in Section 2.5. Note, that the results for "Show, Attend and Tell" omit the CIDEr score, because it was not published. CIDEr and Meteor scores for the best of our models both fall short of the current state of the art. However, our purpose was not to optimize the image captioning model with extra components, but instead we wanted to use a simple enough model that could be easily transferred into a hierarchical RNN.

We also discovered that models with larger vocabulary sizes perform better, and that the best models for image captioning all used a concatenation of DenseCap and ResNet-152 image features. From now on, all the other experiments we did used the vocabulary of the dataset on which the WordRNN was pre-trained and the concatenated image features.

In their "Regions-Hierarchical" model, Krause et al. [30] pre-trained the WordRNN on the Visual Genome Regions dataset. To replicate their model we built support into DeepCaption for training captioning models on the VG Regions dataset by treating each region caption as a standalone alternative caption for the entire image.

*Here, and in all the subsequent tables showing our results, we have used CIDEr-D version of CIDEr to calculate the scores for our models. We have observed that the version of CIDEr typically used in the published research on image captioning is CIDEr-D, however the authors usually refer to it as simply CIDEr, which is what we do in our work as well.

We pre-trained the VG Regions based models with the concatenation of max-pooled DenseCap and ResNet-152 features. We trained the models using the LSTM and the GRU-based RNN implementations, and using $E = 256$ and $E = 1024$ as the embedding size, resulting in four total versions. We picked the best models matching each of the four combinations to be used as the WordRNNs for paragraph captioning. When training image captioning models using the Visual Genome Regions dataset it does not appear to be sensible to use standard Meteor and CIDEr metrics, because the ground truth captions describe the regions, and not the whole image. Therefore, we used the validation loss as an indicator of how good each model was.

The time to convergence measured in epochs was similar for the models trained on MS COCO Captions and VG Regions, and ranged between around 15 epochs for models trained on MS COCO and 20 epochs for models trained on VG Regions. When treating each (region caption, image) pair as a standalone example, the VG Regions dataset contains $3,684,063$ training pairs compared to $414,113$ training examples in the MS COCO Captions. The MS COCO based models took at most a couple of hours to converge, while the ones that used VG Regions required close to 24 hours. Using pre-computed image features benefited us greatly, as training a VG Regions based model with a full underlying CNN would have required up to a week or more per experiment.

The image captioning models trained on both datasets used the "Reduce on Plateau" LR schedule (see Section 4.3.2), with the base learning rate set to 0.001. The learning rate was reduced when validation loss stopped improving for two full epochs. We also tried a static learning rate, as well as different other base rates for the "Reduce on Plateau", which resulted in worse results.

4.4.2 Flat Architecture Results

In order to see how well the flat model performs when faced with the more complex task of paragraph generation, we fine-tuned several of the already pre-trained image captioning models described in Section 4.4.1 by training them further using the Stanford-Paragraph dataset. Each paragraph was concatenated and fed to the model as a single caption. The period character (".") at the end of each sentence was treated as yet another token inside the caption. We set the maximum allowed length for the generated caption to 80 tokens, with longer captions being truncated. All of the flat paragraph captioning models used the LSTM-based RNN implementation.

Our flat models were pre-trained either using the MS COCO Captions or the VG Regions datasets. Each model uses the vocabulary of the dataset it has been pre-trained on – $9,957$ words for MS COCO and $19,804$ words for VG Regions, respectively.

Table 4.5: Comparison of flat models used for paragraph captioning. Our models used the maximum pooled DenseCap features, unless otherwise specified. The score in the top section of the table is from the published paper, while the scores for our models are shown in the bottom section.

Model	**C**	**M**	**B1**	**B2**	**B3**	**B4**
Image-Flat [26, 30]	11.06	12.82	34.04	19.95	12.20	7.71
Flat, VG Regions, $E = 256$	19.00	14.82	37.29	21.85	**13.03**	**7.77**
Flat, VG Regions, $E = 1024$	17.84	**14.98**	**37.74**	21.87	12.77	7.40
Flat, MS COCO, avg-pooled DenseCap, $E = 256$	18.64	14.73	36.84	21.52	12.73	7.57
Flat, MS COCO, $E = 1024$	**20.17**	14.95	37.59	**21.97**	12.98	7.66

Table 4.5 shows the results for our best-scoring flat models used for generating paragraph captions, as well as the current non-hierarchical state of the art, "Image-Flat" [26]. Apart from CIDEr (C) and Meteor (M) scores, we also show BLEU 1-4 (B1-4) results. The model with the highest CIDEr score (marked **bold**) was trained for a total of 77 epochs. Among our flat models, there was not one model which would outperform all others on every single metric, so we chose to highlight (in **bold**) the best scores on per-metric basis.

After checking captions generated by different models, it became apparent that the model with the highest CIDEr score produced the best captions. After analyzing the output generated by different models, we decided to prefer models with the highest or very high CIDEr scores for further evaluation, whenever we are forced to pick which metric to follow.

Interestingly, the results shown for "Image-Flat", typically cited in recent papers on hierarchical paragraph captioning, are obtained from the model by Karpathy et al. [26], similar to the flat model architecture which we are using. This baseline model scores much lower than the architecture we trained. Apart from differences in configuration such as the choice of optimizer – Karpathy et al. used RMSprop, we are using Adam, there are also differences in input features. The earlier model used input features calculated using VGG [59], which is a slightly older model trained on the ImageNet [56] task, similarly to ResNet-152 which we are using. We also use DenseCap features, thus further improving the performance of the model. Therefore, it is evident that more advanced input features available to us today allow for the same model to achieve better scores, with minor changes to training parameters.

In contrast to image captioning models, we see the benefit of using maximum pooled DenseCap features compared to average-pooling. All the models shown in Table 4.5 were trained using a static learning rate. In our experiments, the models that were trained with the "Reduce on Plateau" (see Section 4.3.2) scheduler performed significantly worse. Our results indicate that relying purely on validation loss for learning rate reduction does

not work as well on paragraph data as it did on shorter, sentence length captions. In fact, the validation loss was increasing while the Meteor and CIDEr scores kept decreasing for tens of epochs, confirming a similar finding by Krause et al. [30].

4.4.3 Hierarchical Architecture Results

Having seen the performance scores coming from the flat model, it is now time to introduce the division of labor between SentenceRNN and WordRNN, as described in Section 3.2. Our hierarchical model follows quite closely the "Regions-Hierarchical" model described by Krause et al. [30]. However, there are likely to be differences in implementation because we were not able to obtain the source code of the original model. All our models use concatenation of DenseCap and ResNet-152 features as input. We experiment with WordRNNs trained on MS COCO Captions and VG Regions datasets, unlike the prior work which uses the VG Regions only. The published "Regions-Hierarchical" model uses an embedding size $E = 1024$ and maximum pooling for their DenseCap inputs. In our experiments, we also looked at an alternative embedding size of $E = 256$, and average pooling for the DenseCap image features.

In Table 4.6 we show the results for the best models that use the WordRNN weights pre-trained on MS COCO Captions and VG Regions. Interestingly, we got the best CIDEr score from the model whose WordRNN was first pre-trained on VG Regions and then fine-tuned some more on MS COCO Captions (see the last row in the Table 4.6), so strictly speaking the comparison of this model to the rest is not fully fair. However,

Table 4.6: Comparison of hierarchical models for paragraph captioning. Our models used the maximum-pooled DenseCap features, unless otherwise specified. Results for "Regions-Hierarchical" model are taken from the published paper [30], the rest of the results are ours.

Model	**C**	**M**	**B1**	**B2**	**B3**	**B4**
Regions-Hierarchical [30]	13.52	**15.95**	**41.9**	**24.11**	**14.23**	**8.69**
Hierarchical, MS COCO, $E = 1024$	16.99	15.09	40.09	22.19	12.33	6.79
Hierarchical, MS COCO, avg-pooled, $E = 256$	17.79	15.11	39.84	22.19	12.38	6.91
Hierarchical, VG Regions, $E = 256$	15.87	15.18	38.38	21.63	12.33	7.02
Hierarchical, VG Regions, $E = 1024$	17.01	15.15	39.65	22.33	12.62	7.14
Hierarchical, VG Regions, GRU, $E = 256$	17.49	15.15	39.69	22.14	12.46	7.02
Hierarchical, VG Regions+MS COCO, $E = 1024$	**17.85**	14.88	39.33	21.82	12.21	6.78

the difference was not big compared to some of the other models whose WordRNN was pre-trained on a single dataset. While running the experiments, we also noticed that repetitive use of early-stopping + fine-tuning with manually changed learning rates while using the "Reduce on Plateau" scheduler tied to validation loss did not yield as good results compared to using a single static LR for the entire duration of the training.

We also trained some of the models using GRU RNN: one such model is shown in Table 4.6. By comparing the results with the other models we saw that the choice of the RNN cell implementation did not have a large effect on the results. The number of epochs needed for the CIDEr and Meteor scores to peak did not differ significantly between the models using GRU and LSTM.

When training the models which used WordRNN pre-trained on VG Regions we tried a static as well as cyclical learning rates. All of the highest scoring models in this batch of experiments used a static learning rate of 0.0004. It is possible that by changing the hyperparameters used for cyclical LR, it would have been possible to have better results with this kind of learning rate scheduler as well.

The best model overall was "Regions-Hierarchical" by Krause et al. [30], with an exception of CIDEr in which our models outperformed the baseline. However, it is unclear whether the same version of CIDEr was used to evaluate our models and the published baseline. As mentioned before, we did not have access to the source code of the baseline architecture, and the possible reason for our lower overall scores are the differences in implementation.

4.4.4 Hierarchical-Coherent Architecture Results

The hierarchical-coherent paragraph captioning model, introduced in Section 3.4, is the most complex of the ones we ran our experiments on and it is based on the recently published "Diverse-Coherent" [9] model. As was discussed earlier, the authors of that model believe that the improvements in their model compared to the hierarchical baseline add coherence to the sentences within a paragraph. This improvement can also be interpreted as increased fluency (see Section 4.2). Here, we will examine how our hierarchical-coherent model scored on standard metrics, and in Section 4.4.6 we can also see the captions generated by the model.

When training our model, we chose to use only 1024-wide embeddings, because our previous experiments with the hierarchical model showed that the use of wider embeddings gave us slightly better results. In addition, all of the models we tried used maximum-pooled DenseCap input features together with ResNet-152. We used the same DenseCap feature configuration in order for our results to be comparable to the "Diverse-Coherent" baseline. Table 4.7 shows both the baseline and the results for the three best hierarchical-coherent

Table 4.7: Comparison of hierarchical-coherent models for paragraph captioning to the "Diverse-Coherent" baseline. The score in the top section of the table is taken from the published paper [9], while the scores for our models are shown in the bottom section.

Model	C	M	B1	B2	B3	B4
Diverse-Coherent [9]	**19.95**	**17.81**	**42.12**	**25.18**	**14.74**	**9.05**
Hierarchical-Coherent, cyclical-lr	16.29	15.58	41.07	23.24	13.20	7.45
Hierarchical-Coherent, static-lr, GRU	17.46	15.29	40.52	22.73	12.71	7.04
Hierarchical-Coherent, cyclical-lr	19.23	15.45	40.80	22.96	12.89	7.12

models we trained.

When using the hierarchical-coherent architecture our results showed an improvement over the plain hierarchical model, however, the scores came short of matching the published baseline [9]. There are likely some differences between our and the baseline implementation, based on what we could ascertain from the partial source-code published by the authors, discussed in Section 3.4.1. In addition, the published source code covers only parts of the training process, with the exact details being open to interpretation.

All of our hierarchical-coherent models used WordRNNs pre-trained on VG Regions, since the hierarchical models pre-trained on VG Regions proved to produce slightly better captions. In the hierarchical-coherent experiments we avoided the use of the "Reduce on Plateau" scheduler for the learning rate, and tried cyclical and static learning rates instead. The best model in terms of CIDEr was trained with a cyclical learning rate schedule, showing that this training regimen was more beneficial for training the hierarchical-coherent model than for the regular hierarchical model.

While the majority of our experiments were performed using LSTM RNNs, we also got comparable results from the GRU-based models that we trained. The convergence times to achieve comparable scores were also similar for both types of RNN used, indicating that LSTM and GRU based language models in hierarchical contexts are quite similar in terms of their overall behavior. When inspecting the output of the different similarly scoring models, it appeared that the LSTM-based models generate slightly more accurate captions.

4.4.5 Comparison with State of the Art

In Table 4.8 we compare our paragraph captioning results to the current published state of the art, the best overall model and respective scores are shown in **bold**, as before. For our comparison we chose published paragraph captioning models covered in Chapter 3. The first baseline, "Image-Flat" [26] does not use a hierarchical model. The other baselines

Table 4.8: Comparison with paragraph captioning state of the art. The scores in the top section of the table are taken from the published papers, the scores for our models are shown in the bottom section.

Model name	**C**	**M**	**B1**	**B2**	**B3**	**B4**
Image-Flat [26, 30]	11.06	12.82	34.04	19.95	12.2	7.71
Regions-Hierarchical [30]	13.52	15.95	41.9	24.11	14.23	8.69
RTT-GAN [35]	20.36	18.39	42.06	25.35	14.92	9.21
Diverse-Coherent [9]	19.95	17.81	42.12	25.18	14.74	9.05
Diverse-Coherent (with VaE) [9]	**20.93**	**18.62**	**42.38**	**25.52**	**15.15**	**9.43**
Human [30]	*28.55*	*19.22*	*42.88*	*25.68*	*15.55*	*9.66*
Flat (ours)	20.17	14.95	37.59	21.97	12.98	7.66
Hierarchical (ours)	17.67	15.07	39.51	22.06	12.32	6.8
Hierarchical-coherent (ours)	19.23	15.45	40.8	22.96	12.89	7.12

are all hierarchical, and they use an embedding size of $E = 1024$ and max-pooled DenseCap features for their input. "Regions-Hierarchical" by Krause et al. [30] is the original hierarchical paragraph captioning model; "RTT-GAN" by Liang et al. [35] introduces a generative-adversarial framework to the task of paragraph captioning. In addition, we show the results from two related models by Chatterjee and Schwing both of which use GRU-based RNN. "Diverse-Coherent" without VaE is the model that our hierarchical-coherent architecture is based on, the second model by the same authors extends the architecture by adding VaE-based formulation for producing more diverse paragraph captions. This VaE-based "Diverse-Coherent" baseline is the current paragraph captioning state of the art at the time of writing. Finally, the "Human" baseline was obtained by Krause et al. [30] by having humans generate paragraph captions for a set of 500 randomly chosen images.

For each of the three model types we compare, we chose the best performing one based on their CIDEr and Meteor scores as well as comparing their output captions. Our chosen flat model is pre-trained on MS COCO Captions using max-pooled DenseCap features, and embedding size of $E = 1024$; the hierarchical model is using the same input features and embedding size, but has a WordRNN pre-trained on VG Regions; and finally, our best hierarchical-coherent model is trained using the same inputs, but unlike our other models, it was trained with a cyclical learning rate.

As mentioned previously, our hierarchical models did not reach full parity with the published baselines, since we did not have access to the full source code for either of the models, which we would have needed for exact replication of the published results. At the same time, our flat model outperformed the existing flat baseline on all standard metrics except BLEU-4, where it came close, showing that there may still be a potential for

improvement when using a non-hierarchical model for paragraph captioning. Among the models we trained, the best scores were shared evenly by our flat and hierarchical-coherent models.

4.4.6 Human Evaluation

We saw that among the three architectures we experimented on, the best CIDEr score was reached by the flat model. At the same time it got a slightly lower Meteor score when compared to hierarchical models. How would a human judge the relative performance of these models? In order to answer this question in-depth, we would need to perform costly and time-consuming full human evaluation of all the captions generated by our models. One way to perform this could be via the use of a crowd-sourcing platform, such as Amazon Mechanical Turk*.

However, even without resorting to crowd-sourcing, one can get an initial idea of the output "style" that different models show, as well as the kinds of mistakes they make, by inspecting a few dozen captioned images. In this section we show a small selection of images and their generated captions to give the reader a general idea of the quality produced by different models. Figures 4.3 – 4.5 show captions generated by our models seen in Table 4.8. Figure 4.3 shows examples of images where hierarchical and hierarchical-coherent models perform at least as good or better than the flat model; Figure 4.4, on the other hand, shows images where hierarchical models do not outperform the simpler flat model; and finally Figure 4.5 shows example captions where different models have widely varying output; in the same figure, column (a) we can also witness an excellent caption produced by the hierarchical-coherent model, showing the potential of this architecture.

When assessing the captions, we used color-codes to indicate their perceived quality. We focused on the fidelity of the captions as our primary criterion. The secondary criterion was fluency. Both of these criteria were described in Section 4.2. Fidelity aims to assess how well the generated captions describe the salient details present in the image. Fluency, on the other hand, measures how well the language of the caption "flows". We used **green color** to indicate that the caption has captured most of the salient details correctly; **yellow color** for captions that got some of the salient details correct; and **red color** for captions that got none or very little of the important details correct. We observed that the majority of the captions our models produced had a large enough number of factual mistakes, which is a reason for choosing fidelity as our primary criterion, since improved fidelity would result in larger caption accuracy. Another human evaluation metric, intelligibility, did not feel that relevant to our task, because in general our RNN-

*https://www.mturk.com/

(a)

(b)

(c)

Flat

A baseball player is standing on a baseball field. he is wearing a blue and red uniform and a black and white uniform. there is a man in a blue shirt and black pants with a red and white baseball bat on his left hand. there is a man in a blue shirt and black pants standing in front of him. there is a man in a blue shirt and black pants standing behind him

A yellow and white train is on the tracks. there is a platform next to the train with a roof over it. there is a yellow and blue sign next to the train. there are trees next to the train.

A man is standing in a room. he is wearing a white shirt with a black tie. he is wearing a black tie and blue tie. he is standing next to a large window with white blinds on it

Hierarchical

There is a baseball player in a white and red uniform. The baseball player is in motion playing the game. There are two men on the baseball field wearing white uniforms. There are people sitting in the stands watching the game behind them. There are several empty chairs in the stadium although there are several people watching the game. There are several players sitting behind the field watching the game.

A train is on the train tracks. The first train is painted yellow and blue. The front of the train is yellow with a number 2 on the front. The platform is empty. There is a tall building to the right of the train. There is a yellow line to the left of the train.

A man is standing in front of a building with a white shirt on and a black tie. He is wearing a white button down shirt and a black tie. To the left of him there is a gray wall with a small plant growing on it. The door is white and the walls are made of wood.

Hierarchical-Coherent

A baseball player is swinging a bat that is in his hands. He is wearing a black baseball glove. A baseball player is standing on the field. He is wearing a gray uniform and a black helmet. He has a number on the back of his uniform. There are people sitting in the stands behind the dugout.

A yellow train is on the train tracks. The train driver is seated on the left side, and the other is wearing a white tee shirt. There is a silver train on the tracks. The train is blue and yellow. The headlights are on. It is at a train station platform.

A man is standing outside of a building. He is wearing a white dress shirt with a tie and black pants. His eyes are dark and he has a black beard. He is standing with his hand on his front teeth. Directly behind him is a large grey stone wall with a red, orange, blue and green decoration. The man is wearing a light blue collared shirt with a large white stripe and a red tie.

Figure 4.3: Output examples where the hierarchical models perform equal to or better than the flat model.

(a)

(b)

(c)

	(a)	(b)	(c)
Flat	A man is sitting in a suit and has a tie over his mouth. he is wearing a black suit. there is a white suit jacket on the man 's head. he is also wearing a black suit with a white vest and a white collared shirt.	Two men are playing frisbee in a large field. the man on the left is wearing a red shirt, gray hat, black pants and white socks. he has a white frisbee in his left hand, and a black <unk> on his left hand. he is standing on a grassy field. there is a lot of grass in the area. the ground is made up of different color of brown. the	The image is of a city skyline. a large bridge is over a river that has a train on the bridge. the bridge is a light blue color and the water is calm. the water is calm and still. the water is reflecting the sky and the buildings.
Hierarchical	A man is standing in front of a white wall and a mirror with a clear metal frame. The man is wearing a black suit and a white collared shirt with a tie around his neck. The man is wearing a white button down shirt and a black tie. The man is wearing a white button down shirt. His hair is brown and he has a slight smile. He is wearing a white dress shirt with a black tie.	There is a man in a red shirt and gray pants. The player on the field is wearing a red baseball glove. There are two men in the front of the photo with the red cap. <end>. Behind them are green trees and a blue baseball hat. Behind the wall there are several large buildings, mostly brick, but that is covered in graffiti.	There is a train moving on the tracks. There is a red and white stop sign in front of the train. There are long wooden poles holding the power lines which are visible. There are several boats in the water. A row of small skyscrapers are also parked at the foot of the boat. There is a railing around the boats.
Hierarchical-Coherent	A man is sitting in a white chair with his arms propped up. He has a black sweater over a white t-shirt and a dark grey tie. His eyes are glasses and he has a slight smile on his face. There are wood paneling around the boy 's white colored shirt. There is a dark brown framed picture on the wall above the boy. There is a white staircase in the middle of the room.	This image is of a baseball field. The players are wearing uniforms, maroon and white uniforms. They both are wearing socks and cleated shoes. They are both wearing white socks. He is also wearing a white jersey with red lettering. The man also has an orange shirt on with some gray shorts.	The sky is blue with white clouds. The train is long. The train is black with red stripes on the side and it is pulling into the station. The sky above the water is blue. There are poles at the edge of the runway. The sky is partly cloudy and mostly blue.

Figure 4.4: Output examples where the hierarchical models perform equal to or worse than the flat model.

(a)

(b)

(c)

Flat

There is a white refrigerator with many items on top of it. there is a white refrigerator next to the refrigerator. there is a shelf on the wall next to the refrigerator. there is a white refrigerator behind the refrigerator.

A man is holding a red baseball bat. he is wearing a black baseball hat. he has glasses on. there is a white sign behind the man.

A woman is standing in front of a large red and white donut. the woman is wearing a pink and blue striped shirt. she is holding a clear plastic bag in her right hand. she is wearing a pink shirt with black sleeves. she is also wearing a black apron over her right. she has a pink and white striped shirt and black pants. there is a large silver shelf behind the woman

Hierarchical

A refrigerator door is open. The shelves are in a different room. The shelves are white. There is also a jug of milk.

There is a man wearing a black shirt and black shorts. There is a man with a baseball bat inside his hand and he is looking at the ball. His shirt is a light color and he has a tattoo on his arm. The man has a black hat on his head and chest. There is a man wearing a red shirt with a black hat on his head and holding a black. He is holding a black and gold book and is sitting above his head.

There are several people standing in a store. There are also two other women, some are holding colorful umbrellas and standing behind each other. There are various <unk> in the background and there are other people standing behind her. There are other young women in the background and she is wearing glasses and holding a donut with a. She is leaning over a counter and has a bunch of things on it. There are pictures on the wall to the right of the woman.

Hierarchical-Coherent

The refrigerator has many shelves in it. There is a shelf below the refrigerator. There are a lot of items on the shelves. There are also a few containers of food. There is also a jug of milk on the fridge. There are many containers of food on the shelves.

A man is standing in front of a baseball bat with a sign on it in front of him. He is wearing a baseball uniform. The jersey the boy is wearing is blue and white. He is standing in a baseball field. There are people behind him sitting on the ground.

A woman is smiling for a picture and wearing a large pink face mask on her face. There is another woman standing behind the first shelf. There are many other items that are mostly orange, but not many items for sale. There are alot of people standing in the background. There are pretty little women standing in the booth.

Figure 4.5: Miscellaneous captioning examples.

based language model is capable of generating sentences that on their own usually make sense, it is their accuracy and the smoothness of transitions between sentences that often have issues. When performing human evaluation of image captions, it makes sense to disregarded adequacy (which judges how close the generated caption is to the ground truth), because this metric is mostly instrumental for automatic image evaluation, while fidelity is more suitable for matching the image content to captions directly, which is what human evaluators would do.

Most generated captions we saw contain small inaccuracies, such as "left" instead of "right"; incorrect item or entity counts – there are often cases where more than three or more entities are present, and the output caption still reads "two" as the corresponding count. The models that describe the clothing of the person shown in the image often mention both the correct garment types and their colors, and then proceeds to list the same garments with different colors. Or as we see in Figure 4.3(c), where the hierarchical-coherent model incorrectly labels the facial hair visible on the man in the picture as a "beard" instead of "mustache". When assigning evaluation color-codes, we were quite lenient towards some of these small factual errors.

When examining a larger subset of images we noticed several patterns in the kinds of images that are easy or difficult for our models to caption. Images taken in low light are difficult for all models to caption correctly. Another problematic area are object close-ups, where a large amount of detail is seen in the background, for example in Figure 4.5(c) the close-up of a woman combined with a rich background seems to confuse the hierarchical models. Images with non-typical context or where the context is not immediately clear cause our models some difficulties as well.

All in all, the generated results were often contradictory and difficult to evaluate. It was often so that some sentences in the same caption used sound language and were factually correct, while other sentences in the same caption were either incorrect or nonsensical.

At the same time, there were also several genres of images where the models performed well. These could be broadly divided into:

- Images depicting humans engaged in outdoors activities such as *baseball*, *surfing*, *skiing*, *frisbee*;
- Images of land vehicles such as close-distance shots of *cars*, *motorbikes*, *trucks*, *trains*, and to a lesser extent also *bicycles*;
- Images showing various animals running or grazing in the grassy plains, such as *zebras*, *giraffes*, *sheep*, *horses*, etc;
- Images of *food*.

In addition, most of the time our best models showed an ability to tell apart *women* from *men*, and *grownups* from *children*. The difference in performance of our models on the different image types is most likely related to the kind of data the models used in the encoder and decoder components were pre-trained on.

4.5 Discussion

In our experiments, we implemented and compared three main model types – flat, hierarchical, and hierarchical-coherent. We also pre-trained our models on two different datasets before further training them on the Stanford-Paragraph dataset. Based on our experience, there was not in fact that much difference between pre-training the models on MS COCO Captions versus VG Regions, however the models pre-trained on the VG Regions seemed to have better BLEU-4 scores. This could be explained by the fact that the Stanford-Paragraph dataset is built on top of the VG Regions, and some of the longer n-grams present in individual region captions have found their way into the paragraph dataset [30]. Using pre-calculated features sped up training for all of our models, and based on our initial experiments, we saw that all models benefited from using pre-trained WordRNN weights as well.

4.5.1 Scoring the Models

We follow Karpathy et al. [26], and use the coco-caption* code to compute the BLEU-{1,2,3,4}, Meteor and CIDEr scores, with CIDEr-D version being produced for the latter. When comparing our results to the published baselines, we rely on the published scores for comparison. There are two main aspects affecting the scores: how the captions were prepared and how the scorers were configured. The first aspect comprises the type of tokenization performed on the captions – such as stemming and punctuation removal, as well as whether lower-casing of model output was done. The scorer configuration includes the configuration parameters for the metrics and the possible differences in the implementation of the code, such as the choice of the CIDEr version. Thus, it is unclear, whether our results are in all cases strictly comparable to the published results.

4.5.2 Flat Versus Hierarchical Models

We saw that our flat model performed well when compared to the two hierarchical models in terms of automatic scores it obtained. Based on our findings, it cannot be ruled

*https://github.com/tylin/coco-caption

out that if more work is put into improving the flat model, it may even reach state of the art performance. The key advantage of the flat model is that it is based on a simpler architecture, which in turn means that it is easier to train and to add still more improvements, such as the ones mentioned in Sections 2.5 and 3.5.

When looking at the generated captions we saw that images in the validation dataset vary quite greatly, some images seem to be easier to caption with paragraph length descriptions than others. The flat model tended to generate shorter captions that often felt like an extension of a single sentence caption. Qualitatively, it seems that our models represent something one might call a "spectrum of richness" of language – starting with simpler descriptions from the flat model, and more elaborate paragraph captions coming from the hierarchical-coherent one. However, it appears that the richness of language does not always mean higher fidelity in describing the actual image, and most of the generated paragraphs we looked at always had some incorrect details.

4.5.3 Potential Improvements

Ultimately, the best captions are the ones which are judged as such by humans. How then can we improve the performance of our models based on human evaluation criteria? A few things come to mind on how the models can be changed to improve on the corresponding metrics:

- Fidelity – by improving the input features, and adding some sort of attention mechanism, where captions are grounded on salient input elements;
- Intelligibility – by having a better underlying language model;
- Adequacy – by directly using caption scoring in the loss function;
- Fluency – already partially addressed by the coherent model, and would likely require a combination of the elements mentioned above to be further improved.

Several other improvements come to mind: using pre-trained ELMO [51] and Bert [15] embeddings to better capture the semantics of the captions; Employing reinforcement learning for optimizing the CIDEr, Meteor and BLEU scores of paragraph captioning output directly, similar to the approach [54] briefly covered in Section 2.5; Tuning the architecture by modifying the hyperparameters that affect training – such as various learning rate schedulers and optimizers, as well as doing small changes to how the model is built and configured.

5. Conclusion

In this thesis we described three different existing architectures that can be used for paragraph captioning. The simplest of the three is the flat model originally developed for image captioning. In its original form it is composed of a convolutional neural network encoder and an RNN-based decoder. The other two models – the hierarchical and the hierarchical-coherent – extend the flat model by adding a top-level RNN for separately keeping track of sentence context. The hierarchical-coherent model extends the hierarchical model by adding a mechanism for allowing the gradients to flow directly from the last word of the previous generated sentence to the next sentence via the coherence vectors.

In our experiments we implemented the flat, the hierarchical and the hierarchical-coherent models. For input, we used both the DenseCap [25] features used in paragraph captioning together with features extracted from ResNet-152 [23]. The models used either LSTM or GRU-based RNNs. Most of the models were trained using the former. However, for the GRU-based models, we did not notice a significant difference to LSTM. We evaluated our models using standard CIDEr-D, Meteor, and BLEU-$\{1, 2, 3, 4\}$ metrics.

While some of our models approached the baselines [9, 30], we did not manage to replicate the published scores for the two hierarchical models that our work was based on. In the case of the flat model, we exceeded the published "Image-Flat" [26] baseline on all the above metrics. In particular, we have shown that with minor improvements to a simple image captioning model by Vinyals et al. [66], one can obtain much higher scores on standard metrics for paragraph captioning using any flat model than were previously reported. In fact, the performance of our flat model measured by the CIDEr-D metric is very close to the current hierarchical state of the art.

It is yet inconclusive whether there is a ceiling on how much a flat model can improve compared to a hierarchical model. In addition, our results do not provide any indication on whether a flat model can be fully competitive with hierarchical models when evaluated by human judges. Our quick empirical evaluation of a few dozen paragraph captions generated by the three models shows that different models have slightly different output "style". In particular, the fluency of the output in hierarchical-coherent models seems

to be somewhat higher than in its flat counterpart, but this is not captured in scores produced by the standard metrics. Therefore, improvements to standard metrics are needed to better align with the human evaluation criteria.

In the future, several of the recent improvements to image captioning, including the use of attention and reinforcement learning techniques, could be applied to both the flat as well as the hierarchical models. For example, using reinforcement learning to optimize the CIDEr and Meteor scores directly may yield an overall improvement to the generative power of the model. In addition, applying attention to the models may allow the individual words and sentences to describe specific regions of the image more accurately. Another promising direction of research is to seek out newer dense captioning models and see if they can produce features which are even better for paragraph generation, compared to the dense captioning model that we used.

Acknowledgments

I would like to thank the people without whom this thesis would not have been what it is: Patrik Floréen for his thorough and helpful guidance during the process of writing this thesis and for providing me with an opportunity to work at the CBIR group at Aalto University on such an interesting topic; Mats Sjöberg for being an excellent and patient mentor and a great colleague – this thesis exists thanks to our collaboration on DeepCaption; Jorma Laaksonen for giving me interesting tasks and gently nudging me in the direction of paragraph captioning; Hamed Tavakoli for sharing his insight during our hallway discussions; Adi, Hector, Julius and Phu for being jolly good colleagues and teaching me with their own example of how to work hard and get things done; and finally, Susanne for diligently reading various versions of the manuscript, and providing helpful comments and feedback.

In addition, I would like to acknowledge the Aalto Science-IT project and CSC - IT Center for Science, Finland, for the computational resources provided.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from: https://www.tensorflow.org/.

[2] A. Alzu'bi, A. Amira, and N. Ramzan. Semantic content-based image retrieval: A comprehensive study. *Journal of Visual Communication and Image Representation*, 32:20–54, 2015.

[3] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6077–6086, 2018.

[4] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, 2005.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003.

[8] Y. Bengio, P. Simard, P. Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[9] M. Chatterjee and A. G. Schwing. Diverse and coherent paragraph generation from images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 729–744, 2018.

[10] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[13] B. Dai, S. Fidler, R. Urtasun, and D. Lin. Towards diverse and natural image descriptions via a conditional gan. *arXiv preprint arXiv:1703.06029*, 2017.

[14] M. Denkowski and A. Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, 2014.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[16] S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems (NIPS)*, pages 493–499, 1996.

[17] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256, 2010.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[20] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[21] J. Gu, Z. Lu, H. Li, and V. O. Li. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*, 2016.

[22] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947, 2000.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[24] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[25] J. Johnson, A. Karpathy, and L. Fei-Fei. DenseCap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4565–4574, 2016.

[26] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3128–3137, 2015.

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[29] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 971–980, 2017.

[30] J. Krause, J. Johnson, R. Krishna, and L. Fei-Fei. A hierarchical approach for generating descriptive image paragraphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3337–3345. IEEE, 2017.

[31] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision (IJCV)*, 123(1):32–73, 2017.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[33] A. Lavie. Evaluating the output of machine translation systems, 2011. https://www.cs.cmu.edu/~alavie/Presentations/MT-Evaluation-MT-Summit-Tutorial-19Sep11.pdf, accessed 2019-03-19.

[34] J. Li, M.-T. Luong, and D. Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015.

[35] X. Liang, Z. Hu, H. Zhang, C. Gan, and E. P. Xing. Recurrent topic-transition GAN for visual paragraph generation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3362–3371, 2017.

[36] R. Lin, S. Liu, M. Yang, M. Li, M. Zhou, and S. Li. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907, 2015.

[37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[38] J. Lu, C. Xiong, D. Parikh, and R. Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 375–383, 2017.

[39] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[40] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[41] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[42] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[43] G. A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[44] J. Mitchell and M. Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429, 2010.

[45] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[46] National Research Council (US). Automatic Language Processing Advisory Committee. *Language and machines: Computers in translation and linguistics; a report*, volume 1416. National Academies, 1966.

[47] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[48] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[49] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.

[50] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[51] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[52] D. M. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.

[53] Y. Pu, W. Yuan, A. Stevens, C. Li, and L. Carin. A deep generative deconvolutional image model. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 741–750, 2016.

[54] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017.

[55] S. Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520, 2004.

[56] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[57] R. Shetty, M. Rohrbach, L. A. Hendricks, M. Fritz, and B. Schiele. Speaking the same language: Matching machine to human captions by adversarial training. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[58] R. Shetty, H. R. Tavakoli, and J. Laaksonen. Image and video captioning with augmented neural architectures. *IEEE MultiMedia*, 25(2):34–46, 2018.

[59] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[60] M. Sjöberg, A. Polis, A. Surikuchi, and J. Laaksonen. DeepCaption: deep learning-based image captioning framework, 2018–2019. Open-source software https://github.com/aalto-cbir/DeepCaption.

[61] L. N. Smith. Cyclical learning rates for training neural networks. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[62] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014.

[63] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112, 2014.

[64] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[65] R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575, 2015.

[66] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2015.

[67] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(4):652–663, 2017.

[68] L. Wang, A. Schwing, and S. Lazebnik. Diverse and accurate image description using a variational auto-encoder with an additive gaussian encoding space. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5756–5766, 2017.

[69] J. White, T. O'Connell, and F. O'Mara. The ARPA MT evaluation methodologies: Evolution, lessons, and future approaches. In *Proceedings of the First Conference of the Association for Machine Translation in the Americas*, pages 193–205, 1994.

[70] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[71] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, pages 2048–2057, 2015.

[72] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*, pages 3320–3328, 2014.

[73] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4584–4593, 2016.

Appendix A. Human Evaluation Examples

In the following tables we show human evaluation scores for paragraph captions generated by seven different models, using the human evaluation criteria of fidelity (fid), intelligibility (int), adequacy (adq), and fluency (flu), covered in Section 4.2.

(a) id: 2366857

(b) id: 2370296

(c) id: 2412624

(d) id: 2386694

(e) id: 2415464

(f) id: 2371418

Figure A.1: Six images sampled from the Stanford-Paragraph validation dataset.

Table A.1: Ground truth and generated captions for image id: 2366857.

Model	Caption	Fid	Int	Adq	Flu
Ground truth	A man in the foreground is holding a kite with Spiderman on it. The tail of the kite is red and is waving in front of the man. He is wearing a pink shirt and a striped tie. He has sunglasses on and a black hat. People behind him are on a lawn also flying kites that are visible in the sky. A bridge is in the distance to the right and cars are in the parking lot next to the vast lawn.				
Flat	A man is holding a kite. he is holding the kite in his hand. the man is wearing a red jacket.	2	3	3	3
Hierarchical Coco	This picture is taken outside on a cloudy day. She is wearing a long sleeve blue shirt and a blue skirt. The kite is colorful. The man is wearing a red cap, a long sleeve blue shirt and blue jeans. Tall buildings can be seen at the back of the field. A large building can be seen at the edge of the field.	1	2	2	2
Hierarchical VG Regions	A man in a black shirt and blue jeans is walking towards the field. The sky is blue and clear with a few clouds. The sky is mostly cloudy and overcast. There are buildings in the background of the photo. There are two tents set up on the grass in front of the lawn. <end>.	1	2	1	2
Hierarchical VG Regions	A man is flying a kite on the beach. He is wearing a short sleeve shirt and shorts. He has on a black sunglasses and a black and white t shirt. He has long hair and is wearing glasses. Many of the homes have many different colors on them. Some of the kites are colorful.	3	3	2	2
Hierarchical Coherent	There are people standing in the grass on a field. There is a person flying a kite that is flying in the air. The sky is mostly cloudy with mostly white clouds in the sky. Behind the kite there are people sitting in the grass. There are many people standing in the grass and looking up. The sky is mostly cloudy.	3	3	2	2
Hierarchical Coherent	There is a man standing in a field flying a kite. The kite is red and black. There are two people standing in the grass. There is a red kite in the sky that is in the air. There are trees in the background. There is a man in a red shirt and a red cap on while the man is wearing a.	2	2	2	2
Hierarchical Coherent	This picture is taken outside on a cloudy day. A man is wearing a red jacket and black pants is flying a kite in the sky. The kite is rainbow colored and is in the shape of a bird. There are two people standing in the field watching a kite fly. There are several kites in the sky above the people. The sky above is a light blue with sparse white clouds in it.	3	3	3	3

Table A.2: Ground truth and generated captions for image id: 2370296.

Model	Caption	Fid	Int	Adq	Flu
Ground truth	"A large jet passenger plane that has the company name ""IBERIA"" on it. There is a ""B"" emblem on the tail and it is colored yellow and and dark orange. The side engines are visible and the jet plane is located on the runway of the airport. There are white stripes on the runway and a building in the back ground. "				
Flat	A large white and blue airplane is parked on a tarmac. there is a large white and red truck in the background. there are two sets of double doors. there is a large building to the right of the plane. the sky above the buildings is overcast.	2	3	2	2
Hierarchical Coco	The sky is partly cloudy. The lights are shining on the inside of the bus. The lights are on. There are lights on the ceiling of the bus. There are lights on the ceiling of the building. The sky is clear and blue.	1	1	1	2
Hierarchical VG Regions	There is a white airplane in the sky. There is blue and white stripe under the windows and the door of the plane. There is one light shining brightly. There are many lights on the bridge that are hanging over the street. There is a light on one of the lights around the lights. <end>.	2	2	1	1
Hierarchical VG Regions	On the plane is a jet. There are white smoke coming out of the windows. The plane is white and blue. The sky is partly cloudy. The lights on the top of the airplane are shining on ground. The sky is white.	1	1	1	1
Hierarchical Coherent	A white and blue train is sitting on the tracks. There are headlights on the front of the truck. There is a sign below the clock that says no bears. The bus is driving in a tunnel which is covered with passengers. The lights are on in the tunnel. The street lights are red.	1	1	1	1
Hierarchical Coherent	A large white and red airplane is flying over the terminal. The lights are lit up at the top of a light pole at the top of the highway. The lights are on and bright green in color. Below the lights there is a bridge that has an overhead lights. Below the lights is a bright white building with many windows. Behind the lights there is a lighted building, but that is an illuminated yellow color.	2	1	2	1
Hierarchical Coherent	A large white and black commercial bus is driving down the street. The sky is overcast and the sun is setting. The lights are bright and shining brightly. There is also a yellow line on the concrete of the platform. There are many red lights around the lights and on the front of the building. There is also a number of green and yellow letters on the side of the train.	1	1	1	1

Table A.3: Ground truth and generated captions for image id: 2412624.

Model	Caption	Fid	Int	Adq	Flu
Ground truth	a group of people are ordering some food. the people are all dressed well. the sign on the side is yellow. the sign has pictures on it.				
Flat	People are standing inside of a store. the store has a lit up top for sale in the store. the front door of the store has a black metal shelf with <unk> of liquor bottles on it as well as other business items. there are white canopies, including white paper with red chinese on them. the tables have white <unk> on them.	1	1	1	1
Hierarchical Coco	A man in a business suit is talking on a phone 's display 's neck. There is a man standing in the middle of the image. The man is wearing a black long sleeve shirt and a tie. The building behind the people is also. The sky is bright and blue. The sky is lit up with the sun shining.	1	1	1	1
Hierarchical VG Regions	A man is standing in front of a store. There are a few customers in front of him, and they are all wearing black coats. There are all customers in front of him. There are all customers in the store. There are many signs hanging from the ceiling in the back. They are all wearing robes and some have pink.	2	1	1	1
Hierarchical VG Regions	A group of people are inside a store. The people are all taking pictures from a sign. The display for <unk> is lit up in lights. The display for display is only where the seats are wet. A sign for live music on sunday 's at 7:30 pm. The subway station sign have arrows on them.	1	1	1	1
Hierarchical Coherent	The people are standing in a bar. Most of the people are wearing glasses and no shirt. The man is looking at the phone and trying to it 's hands. She is holding a phone up to her ear and looking at the camera. There are hanging from the ceiling, she holds a white sign and looking at it. There are a couple of people looking at them from the <unk> looking at their time.	1	1	1	2
Hierarchical Coherent	There are people standing inside of a building. There are many men standing and talking to the women. Some men are looking at a cell phone. There is a woman standing behind the man wearing a black shirt and a red scarf around her neck. There are many men standing near the man. The man and woman are standing in the front of the store.	1	3	1	2
Hierarchical Coherent	There are people standing under a large umbrella. There are signs hanging from the ceiling. There are men and women standing in the background. The men are walking around and talking. The men are looking at food and drinks in front of the fridge. There is a customer standing by the man.	1	3	2	3

Table A.4: Ground truth and generated captions for image id: 2386694.

Model	Caption	Fid	Int	Adq	Flu
Ground truth	There are small bushes in a field. There are white sheep in the field eating the grass. There is a large hill behind them. There is tall weeds and grass in it.				
Flat	A large field of grass with some sheep grazing on it. there are hills off in the distance. there are many trees off to the side. there is a fence to the right of the field.	3	3	3	3
Hierarchical Coco	There are three sheep standing in a field. There are white clouds in the sky. There are trees surrounding the field. The water has <unk> from the tree as well. The trees are a little <unk> and others are mostly <unk>. The sheep are light brown and the same is visible.	2	2	3	2
Hierarchical VG Regions	There are two sheep standing together. There are white sheep grazing in the field. There is a small fence to the right of the sheep. A horse is grazing in the grass. Beyond the field are trees, bushes and grass. Trees surround the field.	3	3	2	3
Hierarchical VG Regions	There are several sheep in a field. The sheep are white with dirty fur. There is a white and brown sheep in the middle of a field. There are white sheep standing in the field. The sky is gray and cloudy. The sky is cloudy and gray.	2	3	2	2
Hierarchical Coherent	The picture was taken in black and white. There are a lot of white flowers in the field. There are sheep grazing in the grass in the distance. The sheep are looking directly in the distance. There are a few white clouds in the sky above the scene. The sky is pretty clear and has a bunch of fluffy clouds.	2	3	2	3
Hierarchical Coherent	A house is in the background with a lot of green trees. The sheep are all white. One of the sheep has it 's head down and one is standing up. The sheep are fluffy and grazing. The sheep are fluffy and fluffy. There are trees in the distance, and there are more trees in the background.	3	3	2	2
Hierarchical Coherent	A sheep and sheep are grazing in a field. The sheep have white fur on their legs and white fur. The sheep also have white fur. There are also sheep grazing on the field. There is a forest that also full of trees with green leafs on them. The sky is blue and there are also a white fluffy clouds.	2	2	2	2

Table A.5: Ground truth and generated captions for image id: 2415464.

Model	Caption	Fid	Int	Adq	Flu
Ground truth	A silver and black laptop is open and on top of a desk in a room. The laptop has a black and yellow square graphic on the screen and a blue backdrop. The table the laptop is sitting on has another laptop on it and a glass of water as well. In the background of the room is a flip, white chalkboard and two circular white recessed lighting on the ceiling. Another bright digital screen is powered on in the center of the room. A podium is in front of the screen with a microphone attached to it.				
Flat	A laptop is open on a laptop computer. the laptop is turned on and it is turned on. the laptop is silver in color. the screen on the laptop is black and has a program on the side with <unk> of the letters all white. there is a black and silver laptop on the right of the laptop. there is a black notebook on the table with a white tablecloth underneath it.	2	1	1	1
Hierarchical Coco	There 's a laptop on the table in the middle of the room. The laptop is black. The laptop on the left has a black screen with a white screen under it. The screen is black in color. There is a projector screen from the right above the laptop. The wall behind the laptops is black.	3	2	1	2
Hierarchical VG Regions	A laptop sits on top of a computer desk. There are two computer screens in the image. A white computer mouse is sitting next to the laptop. A laptop and computer are sitting next to the laptop. Behind the laptops, a long white desk with two computer monitors sit on. Behind the laptops are a small black computer monitor.	1	2	1	2
Hierarchical VG Regions	The image is of a laptop computer, and two black dogs are one on which is white,. The laptop is open and is turned on. There is a laptop computer on the desk, with a black pen in it and a dog in. There is a white keyboard on the desk in front of the computer. There is a paper on the right side of the desk, a pen in the holder above the. On the right side of the desk there is a large window.	1	1	1	1
Hierarchical Coherent	A laptop and laptop are sitting on a desk. There is a silver laptop on top of the table. There is a silver laptop on top of the table next to the laptop. There is a silver laptop on top of the desk as well. There is a laptop on the desk as well. There is a phone and a black keyboard.	1	1	1	2
Hierarchical Coherent	Two laptops are sitting on a table. They are both wearing silver screens. There is also a laptop and a laptop turned on. The desk is brown and wooden. A silver laptop is open and near the desktop screen. There is a white paper on the left side of the table.	2	2	1	2
Hierarchical Coherent	Two computer screens are on a desk. One of the laptop has a kleenex stand right next to a laptop. Behind the laptop is a black keyboard with a white keyboard and the buttons are on. Also a laptop computer is open, and a screen on the table, as well as a bottle. Also on the desk is a glass of water and ice. Also is a bottle of water with a small black bottle with a white label.	2	1	1	1

Table A.6: Ground truth and generated captions for image id: 2371418.

Model	Caption	Fid	Int	Adq	Flu
Ground truth	The photo contains a clear vase with water and flowers in it. The vase is on the floor on a black mat. The mat is in front of the door. The flowers in the vase are orange. The vase is half filled with water. The door behind the vase is blue. The door has a white base around it. The day is not bright in the photo. There are green leaves in the vase with the flowers.				
Flat	A vase of flowers is on the table. the vase is a dark color. the vase is a bit <unk> of flowers. the vase is green and has a picture of a flower on it. the vase is made of glass and has a small stem of water on it. the stem of the vase is black. there is a shadow of the vase that the man is holding.	1	2	2	2
Hierarchical Coco	This is an image of a vase. The plant is small and round. The wall behind the plant is yellow. The wall behind the vase is yellow. There is a lot of sunlight behind the table. The wall behind the bench is green.	2	3	2	3
Hierarchical VG Regions	A vase of flowers sits on a table. The vase is on a table and there is a yellow candle in it. The flowers are yellow, orange and green flowers. There is a white vase by the vase. There is a large white vase on the table. There are three small yellow flowers hanging on the wall behind the vase.	2	2	2	2
Hierarchical VG Regions	A glass vase sit on a wooden floor. Green leaves on top of the yellow flowers. Yellow flowers inside of a vase. Behind the flower is the flowers arrangement inside of the white house house. A yellow flower is in a vase in front of the vase. There is a table behind the vase with some papers, a napkin and a few magazines in it.	2	2	2	2
Hierarchical Coherent	A vase is sitting on a glass table. There is a large vase with flowers in it on the table. The vase is filled with water and flowers. The flowers are pink. There is a shadow on the wall. The wall is black.	1	2	2	2
Hierarchical Coherent	This is a picture of a living room. There is a vase inside of the room near the window. The vase is very large and has a white flower in it. Behind the vase is a window sill. The vase has a flower in it, and a bunch of flowers in front of the window. The floor is a light colored tile, and there is a large white building behind the sidewalk.	1	3	1	2
Hierarchical Coherent	The image is of a black vase. The vase is filled with water. The vase is empty, but there is a glass vase sitting on the top of the table with. The vase is sitting on a wooden table with a white flower in it. Behind the vase is a window with white curtains. Next to the table is a vase of green plants.	1	2	1	2